Proceedings of the Detection and Classification of Acoustic Scenes and Events 2018 Workshop (DCASE2018)

Tampereen teknillinen yliopisto - Tampere University of Technology

Mark D. Plumbley, Christian Kroos, Juan P. Bello, Gaël Richard, Daniel P. W. Ellis, Annamaria Mesaros (eds.)

Proceedings of the Detection and Classification of Acoustic Scenes and Events 2018 Workshop (DCASE2018)

TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

Tampereen teknillinen yliopisto - Tampere University of Technology

Mark D. Plumbley, Christian Kroos, Juan P. Bello, Gaël Richard, Daniel P. W. Ellis, Annamaria Mesaros (eds.)

Proceedings of the Detection and Classification of Acoustic Scenes and Events 2018 Workshop (DCASE2018)

Tampere University of Technology. Laboratory of Signal Processing
Tampere 2018

ISBN 978-952-15-4262-6

Preface

This volume gathers the papers presented at the Detection and Classification of Acoustic Scenes and Events 2018 Workshop (DCASE2018), Woking, Surrey, UK during 19-20 November 2018.

The DCASE 2018 Workshop was the third workshop on Detection and Classification of Acoustic Scenes and Events, organized again in conjunction with the DCASE Challenge. The aim of the workshop was to bring together researchers from many different universities and companies with interest in the topic, and provide the opportunity for scientific exchange of ideas and opinions.

The DCASE 2018 Workshop was organized by the Centre for Vision, Speech and Signal Processing (CVSSP) of the University of Surrey. The associated DCASE 2018 Challenge tasks were organized by researchers at a range of international institutions, including Tampere University of Technology (Task 1: Acoustic scene classification); Universitat Pompeu Fabra and Google, Inc (Task 2: General-purpose audio tagging of Freesound content with AudioSet labels); Queen Mary University of London, University of Toulon, University of Crete and University of Salford (Task 3: Bird audio detection); University of Lorraine, Johannes Kepler University, Inria Nancy Grand-Est and Carnegie Mellon University (Task 4: Large-scale weakly labeled semi-supervised sound event detection in domestic environments); and KU Leuven (Task 5: Monitoring of domestic activities based on multi-channel acoustics).

For this edition of the DCASE 2018 Workshop, 59 full papers were submitted, each reviewed by at least three members of our Technical Program Committee. From these, 44 papers were accepted, 14 for oral presentation and 30 for poster presentation.

The Organizing Committee was also pleased to invite leading experts for keynote addresses:

- Hervé Glotin (Université de Toulon, CNRS, LIS, France)
- Hanna Lukashevich (Fraunhofer Institute for Digital Media Technology, Germany)

The success of the DCASE 2018 Workshop was the result of the hard work of many people whom we wish to warmly thank here. We wish to thank all the authors and keynote speakers, as well as all the members of the TPC, without whom this edition of the DCASE 2018 Workshop would not exist. We also wish to thank the organizers and participants of the DCASE Challenge tasks.

This edition of the workshop was supported by sponsorship from Adobe, Audio Analytic, Cochlear.ai, Google, Huawei and Sound Intelligence. We wish to thank them warmly for their valuable support to this workshop and the expanding topic area.

Mark D. Plumbley
Christian Kroos
Juan P. Bello
Gaël Richard
Daniel P. W. Ellis
Annamaria Mesaros

A MULTI-DEVICE DATASET FOR URBAN ACOUSTIC SCENE CLASSIFICATION

*Annamaria Mesaros, Toni Heittola, Tuomas Virtanen**

Tampere University of Technology, Laboratory of Signal Processing, Tampere, Finland
{annamaria.mesaros, toni.heittola, tuomas.virtanen}@tut.fi

ABSTRACT

This paper introduces the acoustic scene classification task of DCASE 2018 Challenge and the TUT Urban Acoustic Scenes 2018 dataset provided for the task, and evaluates the performance of a baseline system in the task. As in previous years of the challenge, the task is defined for classification of short audio samples into one of predefined acoustic scene classes, using a supervised, closed-set classification setup. The newly recorded TUT Urban Acoustic Scenes 2018 dataset consists of ten different acoustic scenes and was recorded in six large European cities, therefore it has a higher acoustic variability than the previous datasets used for this task, and in addition to high-quality binaural recordings, it also includes data recorded with mobile devices. We also present the baseline system consisting of a convolutional neural network and its performance in the subtasks using the recommended cross-validation setup.

Index Terms— Acoustic scene classification, DCASE challenge, public datasets, multi-device data

1. INTRODUCTION

Acoustic Scene Classification is a regular task in the Detection and Classification of Acoustic Scenes and Events (DCASE) challenge series, being present in each of its editions up until now. The standard setup of the task as a basic multiclass classification problem makes the task easily approachable also for the beginner in this field, resulting in large number of participants in the previous DCASE challenges. In the first three editions of the challenge, the acoustic scene classification task has received the highest number of submissions among the available tasks, with 17 submissions in 2013 [1], 48 submissions in 2016 [2], and 97 submissions in 2017 [3].

Each consecutive edition of the challenge has brought a new and larger dataset than previous edition, facilitating use of recent machine learning techniques using deep neural networks that rely on large amounts of data for training. In 2013, the acoustic scene classification task used a development dataset consisting of 10 acoustic scenes each with 10 examples of 30 s, and an evaluation dataset of the same size [1, 4]. In 2016, 15 scene classes were used, each with 78 examples of of 30 s in the development set, and 26 examples per class in the evaluation set [2]. This dataset offered higher acoustic variability than before through its higher number of classes, recording locations and amount of data, and it was the first suitable for use of deep learning methods.

In DCASE 2017, the acoustic scene classification task was made more difficult by using 10 s audio segments, by re-segmenting the complete data available in 2016 (both development and evaluation sets), having 312 segments of 10 s per scene class. A new evaluation dataset was recorded in similar locations approximately one year later than the development data, containing 108 segments of 10 s per class. The temporal gap between the recordings created an unexpected mismatch in acoustic conditions, causing a significant drop in performance in all systems between development and evaluation sets [3]. Outside of DCASE challenge, there are only few other publicly available datasets for acoustic scene classification, notably the LITIS dataset [5], containing 19 classes and having approximately 25 hours of audio, recorded using a mobile phone; the Defreville-Aucouturier environmental audio dataset [6] with 4 main classes (11 detailed classes) and approximately 4 hours of audio; and the UEA Environmental noise datasets [7] with 10 classes and approximately 4 hours of audio in 2 series recorded with different devices. Of these, only the LITIS dataset has an adequate size for modern machine learning methods.

DCASE 2018 challenge introduces a new dataset for acoustic scene classification, having a number of ten classes and 24 hours of high-quality audio. It has smaller number of classes than data from previous challenges, but it is much larger in size and acoustic variability, having been recorded in multiple cities across Europe. This is the largest freely available dataset to date, comparable in size to the LITIS dataset, but it is the only one having recordings in multiple countries, while all other publicly available datasets (within and outside of DCASE) are recorded within a single country or city.

At the same time, parallel recordings performed with different devices provide additional variability in the channel properties, allowing an additional subtask for studying the classification problem in mismatched conditions. All previous public evaluations have been done in matched conditions, with a single device used for recording all data, including evaluation data, but in actual usage scenarios of the methods, channel mismatch could be encountered through device mismatch or difference in recording conditions. Other publicly available datasets contain audio recorded with only one type of device, with small exceptions (e.g. [7]) that do not permit a large-scale study of mismatched devices. A mismatch usually causes a large drop in performance of machine learning based systems, as noticed in DCASE 2017, therefore this new dataset allows development of techniques that can cope with the mismatch.

This paper presents the subtasks and dataset used for Task 1 in DCASE 2018. Section 2 presents the data recording procedure. Section 3 introduces the task definition and specific details on the subtasks, while Section 4 gives details on the experimental setup, including database statistics for each subtask. Section 5 presents the baseline system architecture and the results obtained on the provided experimental setup, and Section 6 presents conclusions and future work.

2. DATA RECORDING PROCEDURE

The TUT Urban Acoustic Scenes 2018 dataset was collected during February-March 2018, containing recordings of ten acoustic scenes, recorded in six large European cities: Barcelona, Helsinki,

*This work has received funding from the European Research Council under the ERC Grant Agreement 637422 EVERYSOUND.

London, Paris, Stockholm, and Vienna. Acoustic scenes included are: airport, shopping mall (indoor), metro station (platform, underground), pedestrian street, public square, street (medium level of traffic), traveling by tram, bus and metro (underground), and urban park. Each scene class was defined beforehand, and suitable locations were selected based on the description.

For each city and each scene class, multiple different locations were used to record audio, i.e. different streets, different metro stations, etc. For each such location there are 5-6 minutes of audio, recorded in 2-3 sessions of few minutes each, with a small temporal gap between them. The original recordings were split into segments with a length of 10 seconds that are provided in individual files. Recording locations are numbered and used to identify all audio material from the same location when partitioning the dataset for training and testing. The information available in the dataset consists of: acoustic scene class, city, and recording location IDs.

Recordings were made using four devices that captured audio simultaneously. The main recording device consists in a Soundman OKM II Klassik/studio A3, electret binaural in-ear microphone and a Zoom F8 audio recorder using 48 kHz sampling rate and 24 bit resolution. The microphones were worn in the ears, therefore the recorded audio mimics the sound that reaches the human auditory system of the person wearing the equipment. This equipment is further referred to as device A.

At the same time, the audio was captured using three other mobile devices (e.g. smartphones, cameras), resulting in audio recordings of different quality. We further refer to these devices as B, C, and D. All simultaneous recordings are time synchronized using Panako acoustic fingerprinting system [8]. The used mobile devices are the following: device B is a Samsung Galaxy S7, device C is IPhone SE, and device D is a GoPro Hero5 Session. During recording, the Samsung phone was handheld at torso height, the IPhone was worn in a sleeve attached to the strap of a backpack, while the GoPro was mounted on the other strap. The mobile phones recorded single channel audio with a sampling frequency of 44.1 kHz, while the audio recorded by the GoPro is originally stereo, compressed, sampled at 48 kHz.

Two different versions of the dataset were provided for system development, namely TUT Urban Acoustic Scenes 2018, containing only material recorded with device A, and TUT Urban Acoustic Scenes 2018 Mobile, containing material recorded with devices A, B and C. All datasets are freely available. [1] [2]

3. TASK DEFINITION

Acoustic scene classification is defined as labeling one audio sample as belonging to one of predefined classes associated to acoustic scenes. There are labeled example training data available for all the classes, and therefore the task is an example of a supervised classification problem, having a closed set of categories, as illustrated in Fig. 1. In the DCASE 2018 acoustic scene classification task there are three subtasks, offering more variety and degree of difficulty for the task, and at the same time extending the basic task towards real-life applications where there may be mismatch between training and evaluation data recording devices, or there may be different sources of training data available.

[1] TUT Urban Acoustic Scenes 2018: https://doi.org/10.5281/zenodo.1228142, https://doi.org/10.5281/zenodo.1293883

[2] TUT Urban Acoustic Scenes 2018 Mobile: https://doi.org/10.5281/zenodo.1228235, https://doi.org/10.5281/zenodo.1293901

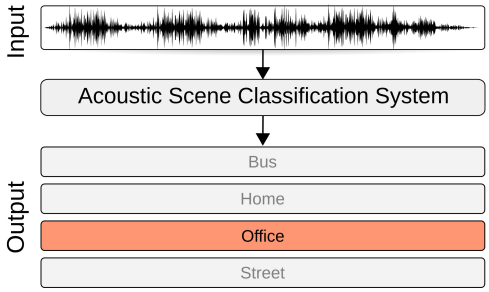


Figure 1: Acoustic scene classification example

Subtask A: *Acoustic Scene Classification* is the typical acoustic scene classification task as encountered previously, where all data, both development and evaluation, are recorded with a high quality device. In this subtask there is no mismatch in recording conditions besides the natural variation of weather, people at the scene, etc, which are not under control of the recorder, but are natural manifestations of the recorded scenes.

Subtask B: *Acoustic Scene Classification with mismatched recording devices* illustrates the problem of creating a system that could be used with multiple devices that record audio of varying quality. In this subtask there is mismatch in audio channels between the development and evaluation sets, which must be accounted for in the system: the training data was recorded with a device providing high-quality audio, while the evaluation data was recorded with multiple devices, resulting therefore in mismatched audio channel. Some amount of parallel data, which was recorded simultaneously with three devices, was also available for training.

Subtask C breaks away from the previous challenge rules against external data sources and allows use of external data and transfer learning to solve the problem provided in subtask A. In subtasks A and B all the participants have the same data available for system development, putting all participants to an equal starting point. In subtask C, systems may be relying on various sources of data, in any form, to study the possible improvement provided by using more data. As a rule for the challenge, the participants were required to use only external datasets that are publicly available for free, and they were also required to inform the organizers about the external data sources for maintaining a list of such resources on the challenge website.

4. EXPERIMENTAL SETUP

The experimental setup is similar for the three subtasks, with the same basic classification problem framed in different ways. In subtask A, only data from device A (high-quality audio) is used, while for subtask B, some data from devices B and C is available as parallel recordings. Subtask C allows the use of external data and transfer learning, but does not provide any additional data, only indicates some sources of data that could be used. For each subtask, a development set was provided, together with a training/test partitioning for system development. Participants were required to report performance of their system using this train/test setup in order to allow comparison of systems on the development set.

The total amount of recorded audio was partitioned into development and evaluation subsets, each containing data from all cities and all acoustic scenes. The development dataset was published when opening the task, provided with full metadata information. The evaluation dataset was published as audio only; the metadata of

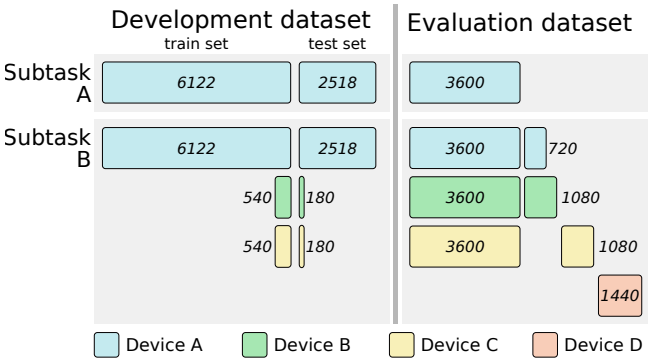


Figure 2: Development and evaluation data amounts.

this part is kept secret, and the evaluation of systems is performed by the task organizers, based on the predicted scene labels that participants have to submit when participating the challenge.

4.1. Development datasets

TUT Urban Acoustic Scenes 2018 development dataset consists of recordings from all six cities, having 864 segments for each acoustic scene (144 minutes of audio). The total size of the dataset is 8640 segments of 10 seconds length, i.e. 24 hours of audio. The dataset is further partitioned into training and test subsets such that the training subset contains audio from approximately 70% of recording locations of each city and each class. Of the total 8640 segments, 6122 segments were included in the training subset and 2518 segments in the test subset. More details on the number of segments from each location are provided in the documentation of the dataset.

TUT Urban Acoustic Scenes Mobile 2018 development dataset contains the same recordings as TUT Urban Acoustic Scenes 2018 and, in addition, two hours of parallel data recorded with devices B and C. Therefore the dataset contains 2 hours of data recorded with all three devices (A, B and C). The amount of data is as follows:

- Device A: 24 hours (8640 segments, 864 segments per scene)
- Device B: 2 hours (720 segments, 72 segments per scene)
- Device C: 2 hours (720 segments, 72 segments per scene)

In this dataset, the data from device A which is originally binaural, was resampled to 44.1 kHz and averaged into a single channel, to align with the properties of the data recorded with devices B and C. The dataset contains in total 28 hours of audio.

The training/test partitioning was done same as for TUT Urban Acoustic Scenes 2018, with approximately 70% of recording locations for each city and each scene class included to the training subset, considering only device A. The training subset contains 6122 segments from device A, 540 segments from device B, and 540 segments from device C. The test subset contains 2518 segments from device A, 180 segments from device B, and 180 segments from device C. The data partitioning is illustrated in Fig. 2.

4.2. Evaluation datasets

TUT Urban Acoustic Scenes 2018 evaluation dataset contains audio examples from locations different that the ones in the development dataset. The dataset contains 3600 segments, therefore 10 hours of audio material, all recorded with device A. In the DCASE

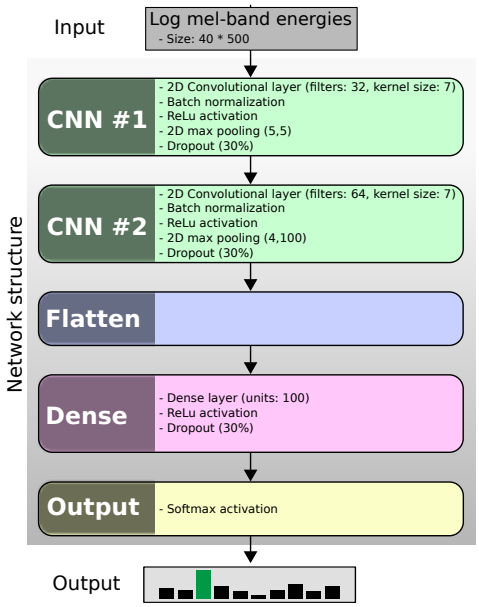


Figure 3: Baseline system architecture.

Challenge, systems will be ranked based on classification accuracy for the evaluation dataset, calculated as class-wise average. The dataset is balanced at class level, having a number of 360 segments per scene, being as balanced as possible at city level too, with 72 segments per scene per city when possible. There are only few exceptions, notably Barcelona airport being available only in the development set.

TUT Urban Acoustic Scenes Mobile 2018 evaluation dataset contains 42 hours of data, recorded with all four devices. The data recorded with device A was resampled and converted to single channel, just as in the development dataset. The dataset contains 3600 segments of parallel data from devices A, B and C, 360 segments of non-parallel data each from devices B and C, and 1440 segments from device D. To create more diversity and prevent guessing of device specific segments, there are an additional 720 non-parallel segments from devices A, B and C, whose sole purpose is to create a non-balanced set, i.e. these segments will not be evaluated.

Ranking of the systems will be done based on classification accuracy only on audio recorded with devices B and C. Data from device A will be used for comparison with subtask A performance, while data from device D, which was not encountered at all in training, will be used to analyze performance on completely unseen devices. No information about device identity was provided with the segments, in order to force generalization and avoid tuning the systems towards the specific devices.

5. BASELINE SYSTEM RESULTS

The baseline system implements a convolutional neural network-based approach (CNN). The architecture of the network is based on one top ranked submission from DCASE 2016 [9], with added batch normalization and changes to layer sizes. This approach aims to implement a popular solution based on previous challenges, and to offer a satisfactory performance for the task.

For each 10-second audio file, log mel-band energies were first extracted in 40 bands using an analysis frame of 40 ms with a 50% hop size. The neural network consists of two CNN layers and one fully connected layer, and uses an input of size 40x500, equivalent

Table 1: Baseline system results for acoustic scene classification, subtasks A and B in DCASE 2018 challenge. In subtask B, ranking is done by average performance on data from devices B and C. Device ID is indicated per column for subtask B.

	Subtask A		Subtask B								
			Development set				Evaluation set				
Acoustic Scene	Dev set	Eval set	B	C	avg (B,C)	A	B	C	avg (B,C)	A	D
Airport	72.9	55.3	68.9	76.1	72.5	73.4	65.8	59.4	62.6	66.8	1.4
Bus	62.9	66.1	70.6	86.1	78.3	56.7	50.5	69.5	60.0	76.1	19.4
Metro	51.2	60.8	23.9	17.2	20.6	46.6	50.2	40.4	45.3	61.9	54.2
Metro station	55.4	52.8	33.8	31.7	32.8	52.9	37.4	44.9	41.2	58.0	65.3
Park	79.1	79.4	67.2	51.1	59.2	80.8	58.6	63.0	60.8	82.9	6.9
Public square	40.4	33.9	22.8	26.7	24.7	37.9	17.0	16.5	16.7	24.3	0.7
Shopping mall	49.6	64.2	58.3	63.9	61.1	46.4	49.2	55.4	52.3	62.2	78.5
Street, pedestrian	50.0	55.3	16.7	25.0	20.8	55.5	35.8	27.3	31.5	53.8	0.0
Street, traffic	80.5	81.9	69.4	63.3	66.4	82.5	69.2	69.9	69.5	83.1	25.7
Tram	55.1	60.0	18.9	20.6	19.7	56.5	41.3	30.9	47.6	63.6	22.9
Average	**59.7** (±0.7)	**61.0**	45.1 (±3.6)	46.2 (±4.2)	**45.6** (±3.6)	58.9 (±0.8)	47.5	47.7	**47.6**	63.6	27.5

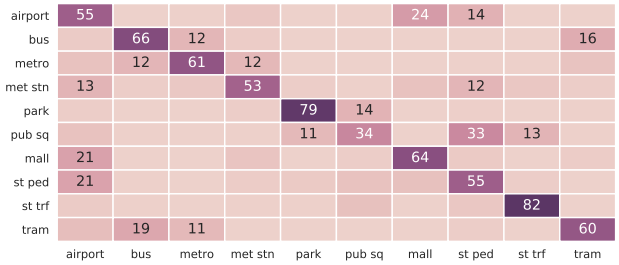


Figure 4: Confusion matrix for subtask A. Evaluation set

to the full length of the segment to be classified. The network was trained using Adam optimizer [10] with a learning rate of 0.001. The system architecture is presented in Fig. 3, including details of each layer. The model selection was done using a validation set consisting of approximately 30% of the original training data, selected such that training and validation sets do not have segments from the same location, and both sets have data from each city. The model performance was evaluated on the validation set after each epoch, and the best performing model was selected.

Table 1 presents the baseline system results for subtasks A and B, both on development set and evaluation sets. In the development stage, the system was trained and tested 10 times using the provided training/test split to account for the effect of random weight initialization in training; the mean and standard deviation of individual performance from these 10 independent trials is presented in the table as development set performance. On subtask A, system performance on the development set is 59.7%, with class-wise results varying from 40.4% to 80.5%. The system generalizes rather well, having a performance of 61% on the evaluation set. By comparing system performance on individual scene classes, we note that most scenes have very similar performance for the two sets - bus, metro station,park street with traffic. The most difficult class to recognize is public square, with the lowest performance in development set and even lower (33.9) in evaluation set, while the best performance of over 80% is obtained for the street with traffic scene. The confusion matrix is presented in Fig. 4.

For subtask B, the system was trained using only the audio material from device A (6122 segments of high-quality audio) to highlight the problem of mismatched recording devices. No additional techniques for dealing with the mismatch was used, in order to avoid influencing the challenge participants in their choice of method. Test subset results are presented separately for each device (2518 segments from device A, 180 segments from device B, 180 segments from device C); average of devices B and C is highlighted, as this is the official ranking measure in the challenge. Here too the system was trained and tested 10 times using the provided training/test split.

System performance on data from device A, same as its training data, is 58.9%, comparable with the performance in subtask A. On devices B and C, the system performs similarly, with an over 10% gap to device A, clearly showing the device mismatch. The system has a similar behavior on the evaluation set, with a performance of 63.6% for device A and 47.6% average on devices B and C, a sign of good generalization and consistent behavior. Performance on device D is very low, with a very large gap even to devices B and C. The one important characteristic of device D is that it provides audio in compressed format, which may be the cause of such extreme mismatch.

Class-wise performance is mostly similar between development and evaluation sets for all devices, with the performance gap still present between devices even when the development and evaluation performance for same device is significantly different: for example metro with 20% (devices B,C) and 46% (device A) in development, increasing to 45% and 61%, respectively, in evaluation set.

6. CONCLUSIONS

The acoustic scene classification within DCASE 2018 challenge offers participants three interesting subtasks, each with own research question. In subtask A, the same classification problem is approached for a dataset with a much larger size and acoustic variability than before, subtask B calls for solutions to the device mismatch problem, while subtask C allows use of external resources such as data and transfer learning for increasing classification performance. The datasets are freely available and not limited for use within the challenge, and will be extended in the future to include more cities and possibly other acoustic scene classes, to further increase task complexity through acoustic variability and allow other challenging research questions, such as training with unbalanced data, or open set classification.

7. REFERENCES

[1] D. Stowell, D. Giannoulis, E. Benetos, M. Lagrange, and M. D. Plumbley, "Detection and classification of acoustic scenes and events," *IEEE Trans. on Multimedia*, vol. 17, no. 10, pp. 1733–1746, October 2015.

[2] A. Mesaros, T. Heittola, E. Benetos, P. Foster, M. Lagrange, T. Virtanen, and M. D. Plumbley, "Detection and classification of acoustic scenes and events: Outcome of the dcase 2016 challenge," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 26, no. 2, pp. 379–393, Feb 2018.

[3] A. Mesaros, T. Heittola, and T. Virtanen, "Acoustic scene classification: an overview of DCASE 2017 challenge entries," in *16th International Workshop on Acoustic Signal Enhancement (IWAENC)*, 2018.

[4] D. Barchiesi, D. Giannoulis, D. Stowell, and M. Plumbley, "Acoustic scene classification: Classifying environments from the sounds they produce," *IEEE Signal Processing Magazine*, vol. 32, no. 3, pp. 16–34, May 2015.

[5] A. Rakotomamonjy and G. Gasso, "Histogram of gradients of time-frequency representations for audio scene classification," *IEEE/ACM Trans. Audio, Speech and Lang. Proc.*, vol. 23, no. 1, pp. 142–153, Jan. 2015.

[6] J.-J. Aucouturier, B. Defreville, and F. Pachet, "The bag-of-frames approach to audio pattern recognition: A sufficient model for urban soundscapes but not for polyphonic music," *The Journal of the Acoustical Society of America*, vol. 122, no. 2, pp. 881–891, 2007.

[7] L. Ma, D. J. Smith, and B. P. Milner, "Context awareness using environmental noise classification." in *INTERSPEECH*. ISCA, 2003.

[8] J. Six and M. Leman, "Panako: a scalable acoustic fingerprinting system handling time-scale and pitch modification," in *5th Conference of the International Society for Music Information Retrieval (ISMIR 2014)*, 2014, p. 6.

[9] M. Valenti, S. Squartini, A. Diment, G. Parascandolo, and T. Virtanen, "A convolutional neural network approach for acoustic scene classification," in *2017 International Joint Conference on Neural Networks (IJCNN)*, May 2017, pp. 1547–1554.

[10] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, 2014.

TOWARDS PERCEPTUAL SOUNDSCAPE CHARACTERIZATION USING EVENT DETECTION ALGORITHMS

Félix Gontier[1], *Pierre Aumond*[2,3], *Mathieu Lagrange*[1],
Catherine Lavandier[2], *Jean-Francois Petiot*[1]

[1] LS2N, UMR 6004, Ecole Centrale de Nantes, CNRS, 44322 Nantes, France, {felix.gontier}@ls2n.fr
[2] ETIS, UMR 8051, Université Paris Seine, Université de Cergy-Pontoise, ENSEA, CNRS, France
[3] IFSTTAR, CEREMA, UMRAE, F-44344 Bouguenais, France

ABSTRACT

Assessing properties about specific sound sources is important to characterize better the perception of urban sound environments. In order to produce perceptually motivated noise maps, we argue that it is possible to consider the data produced by acoustic sensor networks to gather information about sources of interest and predict their perceptual attributes.

To validate this important assumption, this paper reports on a perceptual test on simulated sound scenes for which both perceptual and acoustic source properties are known. Results show that it is indeed feasible to predict perceptual source-specific quantities of interest from recordings, leading to the introduction of two predictors of perceptual judgments from acoustic data. The use of those predictors in the new task of automatic soundscape characterization is finally discussed.

Index Terms— Soundscape, urban acoustic monitoring, event detection

1. INTRODUCTION

The ongoing urbanization process has led to an increase in sound quality concerns. In urban areas the noise has been linked to several health issues including sleep-related troubles as well as heart disease rates, and is a major cause for city dwellers' annoyance in certain areas. In this context, the 2002/49/CE European directive [1] requires that large cities maintain noise maps to facilitate the development of noise reducing plans. These noise maps are mainly based on predictive maps generated using propagation and emission acoustic models. The studies are also 1) often limited to traffic and other transportation sources, and 2) no fusions of simulations with physical measurements are used. Furthermore, the models depend on data that may be at times or in certain locations unavailable or incomplete. The advent of the internet of things (IoT) presents an opportunity for the development of large, scalable networks of acoustic sensors [2, 3]. The "characterization of urban sound environments" (CENSE) project [4] aims at implementing such a network to produce perceptually motivated noise maps.

The ISO 12913-1 [5] standard gives the following definition of soundscape: "the acoustic environment as perceived and understood and/or experienced by people and/or society, in context". The assessment of subjective descriptors [6, 7, 8] such as the liveliness or calmness is thus necessary to evaluate the quality of urban scenes. The relevant attributes describing the appreciation of soundscapes can be mapped in perceptual spaces [9, 10]. The set of considered attributes is reduced to a few dimensions which are used as a basis for perceptual experiments. Specifically, the dimension of pleasantness is increasingly associated with soundscape quality in recent works [11, 12, 13, 14]. Soundscape perception is highly dependent on the composition of the scene [15, 16]. Indeed, each sound source yields a different perceptual response. For example, soundscape pleasantness is likely to be improved by birdsongs and deteriorated by mechanical noises.

Acoustic monitoring applications typically rely on the measurement of energetic (sound levels, eg. L_{Aeq}) and psychoacoustic (eg. Zwicker's loudness N) indicators. These global quantities describe the overall activity, with percentile values linked to event or background assessment. However they do not differentiate sound sources and are thus not sufficient to a perceptual characterization of soundscapes. Additional information about the taxonomic classification of active sources and their distribution in time is needed. Several sets of relevant indicators have been studied [17, 18, 19] to better account for the specificities of each scene and their source composition.

The use of large-scale sensor networks yields a problematic for the extraction of content-related quantities of interest from important amounts of data. Despite a growing interest in the community, machine learning models - to the best of our knowledge - were not yet specifically targeted to the prediction of source-specific perceptual parameters in complex urban environments. Most event detection applications focus on obtaining a precise annotation of source activity, within usual ranges of tens of milliseconds. The estimation of sound levels involves entirely different models through source separation and regression [20] and longer time scales.

We believe that the use of machine listening techniques could greatly benefit the automatic assessment of urban soundscape quality using sensor networks. The aim of this paper is to 1) bring some context of soundscape characterization, and 2) report on a perceptual experiment performed in order to study which features shall be brought by automatic event detection systems in order to gather relevant information for the task of characterize perceptual attributes of the soundscape.

2. SOUNDSCAPE CHARACTERIZATION

Urban soundscape monitoring has only scarcely been studied by the machine listening community[21]. This work aims at contributing to this task by focusing on pleasantness as it is the most recurrent descriptor of urban soundscape quality, though similar studies could be led for other notions such as liveliness.

Several perceptual experiments on the urban soundscape quality have indeed proposed a model of pleasantness from other per-

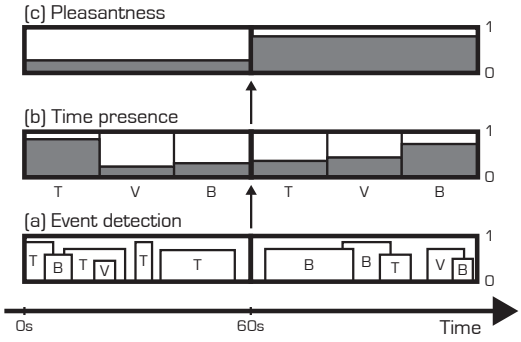


Figure 1: The three suggested levels of metrics to predict soundscape pleasantness. (a) Traffic (T), voice (V) and bird (B) events are detected and their sound level roughly estimated. (b) The perceptual time of presence for each source is computed on one-minute frames, resulting in a pleasantness value (c).

ceptual parameters [22, 9, 14, 13]. In all cases, a good approximation of pleasantness can be obtained by linear combination of both overall and source-specific parameters evaluated on discrete scales. Global parameters consider the sound scene in its entirety for which the overall loudness is commonly used. The parameters used for the assessment of source-wise contributions include 1) the sound level where each source is considered separately, 2) the emergence or dominance relating to the influence of the source in the global mix, or 3) the time of presence, that is the ratio of time where the source s is heard in a given scene. The notion of time of presence is of particularly interest as it hints at the possibility of automatic prediction through event detection systems. The corresponding model is:

$$P = aL + \sum_s b_s T_{s,p} + c \quad (1)$$

where P is the scene's pleasantness, L is the perceived overall level and $T_{s,p}$ is the perceived time of presence for source s. These parameters are evaluated on discrete scales through perceptual tests. The coefficients a, b_s and c are usually found via multiple linear regression and thus differ in each study. Furthermore, three principal source categories are usually identified: mechanical, human and nature. Mechanical sounds are mainly composed of traffic and are mostly found to have a negative impact on soundscape pleasantness, whereas nature sources such as bird activity or water sounds have a positive influence and human sounds (voices) can yield mixed effects.

Assuming this perceptual model, the prediction of pleasantness can be assimilated as that of perceived times of presence of sources. Three levels of metrics are thus identified. First, the physical level Figure 1(a) is evaluated on the presence and emergence of the three identified sound sources: traffic (T), voice (V) and birds (B). The second level Figure 1(b) is the perceived time of presence for each source represented as a scalar in the 0-1 range. The third level Figure 1(c) is the estimate of pleasantness, also represented as a 0-1 scalar. Both the perceptual levels of metrics are only relevant on longer time scales, about one minute being a usual value in existing experiments.

The transition model between the perceived time of presence per source and pleasantness has already been proposed. However no previous work exists that uses detection models for the estimation of source-specific subjective parameters. The feasibility of assessing source perception from the postulated metrics at the physical level shall be verified as a first step prior to building the full estimation model.

3. FROM PHYSICAL TO PERCEPTUAL TIME OF PRESENCE OF SOURCES

We thus conduct a perceptual experiment to validate this key step of the estimation procedure. We wish to study the relation between extracted source-dependent physical indicators to their perceptual equivalents, then validate the relevance of the first level of metrics introduced in the previous section.

3.1. Perceptual Test

For this test, a set of sound scenes recorded in the 13th district of Paris as part of the GRAFIC project [14] is used as reference. Some artificial scenes with equivalent event sequencing are also used for which the acoustic properties of each active source can be computed precisely.

Of the 19 different recording locations, 9 are selected to represent diverse compositional properties: park (P3, P9), quiet street (P5, P11, P13, P17), noisy street (P2, P6) and very noisy street (P16). Corresponding artificial scenes are simulated following the method described in [23]. Simulations are obtained using the *simScene* software [24]. To do so, the recordings are first annotated by identifying active background and event sources. Background sounds are present throughout the whole scene and are characterized by an absolute level parameter. Conversely, events are localized occurrences that are defined by their onset and duration as well as an event-to-background ratio (EBR). The sound scenes are simulated from these annotations and a database of extracts for isolated sources obtained on *freesound.org*, see [23] for more details. This ensures that ground truth source-specific presence and sound level can be computed. One minute of audio is extracted for each scene such as no single event overwhelms the rest of the excerpt.

During the test, the order of appearance is as follows: the original recorded scenes from locations P3 and P16 representing quiet and very noisy environments are always presented first to help participants use the full range of the scale during the test. The 9 simulated sounds are then presented in random order to limit order biases over the participants population. For each scene, 14 criteria are evaluated on a 0-10 scale by the subject. These parameters are displayed in French, but translated in English in this paper for the sake of clarity. The first four questions cover global perceptual parameters:

1. *Noisy - Quiet*: Overall perceived loudness (OL),
2. *Boring, uninteresting - Stimulating, interesting*: Interest (I),
3. *Inert, amorphous - Lively, eventful*: Liveliness (L),
4. *Agitated, chaotic - Calm, peaceful*: Calmness (C).

Source-specific perceived time of presence (scale *Never - Continuously*) and sound level (scale *Very low - Very high*) are also evaluated. The considered sources are traffic (T), birds (B), horns and sirens (H), human voice (V) and footsteps (F). The perceived time of presence and level for source s are respectively noted $T_{s,p}$ and $L_{s,p}$ in the remainder of this paper.

Participants can only listen to each scene once and must answer all questions before proceeding to the next scene. All subjects used the same hardware desktop configuration, sound card and software,

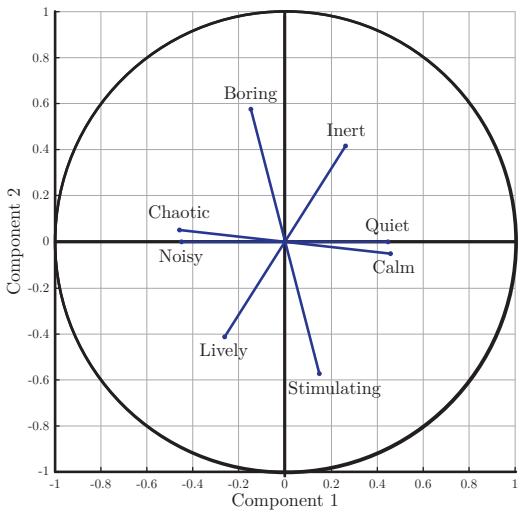


Figure 2: Principal component analysis (first two components) of the four general perceptual parameters at the scene level (n=9). The observed space is distorted although comparable that of previous works in the literature.

as well as Beyerdynamics DT-990 headphones in a quiet environment. The same output volume on headphones was set by the experimenter for all scenes and participants. The resulting playback sound level ranged from approximately 50 dB to 78 dB over the corpus. 30 subjects took the test in 3 sessions, all reported normal hearing conditions.

3.2. Perceptual space

An outlier detection procedure is applied on the 270 resulting assessments (30 subjects, 9 scenes). An assessment is rejected when its distance from the mean is higher than 3 standard deviations at the question level, that is for each parameter of each scene. The results from two participants with more than 10% assessments considered as outliers are removed from the study.

The perceptual space produced by the test is first compared to previous studies in the literature. This is to ensure that relevant conclusions can be made on further analysis. Figure 2 shows the standardized principal component analysis (PCA) of the average values of the four general questions at the scene level (n=9). The first two components respectively explain 52.3% and 30.5% of the global variance. It is found that liveliness (L) correlates poorly with calmness (C), while interest (I) is between the two. These results can be compared to previous studies on similar soundscape qualities parameters [9, 10, 25], where interest and calmness were established as almost independent. The scale of liveliness was in both cases correlated similarly with the two others. However, just as the principal components space is slightly distorted in [25] due to the study being focused on park environments. The correspondence between these results on global perceptual parameters and those of the literature allows us to think that perceptual data on sources are relevant for this study.

3.3. Proposed indicators

As discussed in Section 2 several models have been established to assess pleasantness as a function of global and source-specific parameters. The main objective of this work is to link physical indicators to perceptual source-specific parameters to ultimately predict pleasantness from acoustical data without perceptual assessments. Thus, physical indicators are computed from the audio tracks obtained during scene simulation. To evaluate the overall loudness of the scene, three measurements are chosen in accordance to previous studies [11, 13, 14]:

- L_{50}: Z-weighted (no weighting over the observed frequency range) sound level exceeded 50% of the time in dB,
- L_{A50}: A-weighted sound level exceeded 50% of the time in dBA,
- L_{50} for the 1kHz band only.

Source-specific indicators are also computed: the time of presence and an emergence estimation metric (resp. T_s and L_s for source s), obtained by subtracting the global L_{90} (Z-weighted level exceeded 90% of the time), found to represent well background activity, to the L_{10} of each source. Sound levels are computed with the Matlab ITA toolbox [26] in the 20 Hz-20 kHz range.

In the considered scenes, background sources are always active. The measurement of time of presence is thus limited to sound events which leads to relatively poor representation of the scene perception. Furthermore, the ground truth indicators are computed for each source separately and do not consider the potential impact of other sources active at the same time. Two additional indicators are thus designed regarding these considerations.

The first proposed indicator $T_s(\alpha)$ is a time of presence metric relying on the emergence of each sound source relative to the others. Sound levels (dB) are computed for audio frames of 125 ms. This duration is approximately that of the shortest event found during annotation and corresponds to the "fast" measurements used in acoustical monitoring applications. The emergence, *i.e.* difference $\Delta_s(t)$ of sound levels between the studied source ($L_s(t)$) and the background constituted of all others ($L_b(t)$) is computed. The source is then considered present on a given time frame if the emergence is greater than a threshold value α. A time of presence measurement is obtained by averaging over time:

$$T_s(\alpha) = \frac{1}{N_t} \sum_{t=1}^{N_t} \mathbb{1}_{\Delta_s(t)>\alpha} \tag{2}$$

where N_t is the total number of 125 ms analysis frames in the scene. The optimal threshold α is optimized via grid search to maximize the resulting correlation with the 45 average perceptual time of presence assessments. An optimal value of $\alpha = -31dB$ for the considered corpus is found. As the sound levels of tested scenes range from 50 dB to 78 dB (cf. Section 3.1), only sources with very low sound level on the whole spectrum are considered not heard.

However, the masking of a sound by another does not depend only on the emergence over the whole frequency spectrum. The spectral distribution is important, the level comparison shall thus be made around the characteristic frequency components of a source. A second indicator $T_s(\alpha, \beta)$, based on a spectral decomposition is thus proposed. Third-octave bands sound levels are computed on 125 ms frames and the emergence of a source compared to the background is defined as

$$\Delta_s(t, f) = L_s(t, f) - L_b(t, f) \tag{3}$$

Table 1: Pearson correlation coefficients between perceptual parameters and physical indicators at the scene level (n=9). *: $p < 0.05$, **: $p < 0.01$, non-significant correlations ($p > 0.05$) are noted NS.

Phys./Perc.	OL	I	L	C	$L_{T,p}$	$T_{T,p}$	$L_{B,p}$	$T_{B,p}$	$L_{H,p}$	$T_{H,p}$	$L_{V,p}$	$T_{V,p}$	$L_{F,p}$	$T_{F,p}$
$L_{50,1kHz}$	0.93**	NS	NS	-0.92**	0.75*	0.7*	NS	NS	NS	NS	NS	NS	NS	NS
L_{50}	0.98**	NS	0.73*	-0.97**	0.72*	NS	NS	NS	NS	NS	NS	NS	NS	NS
L_{A50}	0.96**	NS	0.73*	-0.94**	NS	NS	NS	NS	NS	NS	NS	NS	NS	NS
T_T	NS	NS	NS	NS	NS	NS	NS	NS	NS	NS	NS	NS	NS	NS
L_T	NS	NS	NS	NS	NS	NS	NS	NS	NS	NS	NS	NS	NS	NS
T_B	NS	0.67*	NS	NS	0.71*	0.75*	NS	NS	NS	NS	NS	NS	NS	NS
L_B	NS	0.93**	NS	NS	-0.84**	-0.83**	0.91**	0.82**	NS	NS	NS	NS	NS	NS
T_H	NS	NS	NS	NS	NS	NS	NS	NS	NS	0.84**	NS	NS	NS	NS
L_H	NS	NS	NS	NS	NS	NS	NS	NS	0.98**	0.78*	NS	NS	NS	NS
T_V	NS	NS	NS	NS	NS	NS	NS	NS	NS	NS	NS	NS	NS	NS
L_V	NS	NS	0.81**	NS	NS	NS	NS	NS	NS	NS	0.84**	0.88**	NS	NS
T_F	NS	NS	NS	NS	NS	NS	NS	NS	NS	NS	NS	NS	0.9**	0.68*
L_F	NS	NS	-0.72*	NS	NS	NS	NS	NS	NS	NS	-0.69*	-0.78*	0.92**	NS
$T_T(\alpha)$	NS	-0.81**	NS	NS	0.90**	0.94**	NS	NS	NS	NS	NS	NS	NS	NS
$T_T(\alpha,\beta)$	NS	-0.80**	NS	NS	0.88**	0.92**	NS	NS	NS	NS	NS	NS	NS	NS
$T_B(\alpha)$	NS	0.88**	NS	NS	NS	NS	0.95**	0.97**	NS	NS	NS	NS	NS	NS
$T_B(\alpha,\beta)$	NS	0.88**	NS	NS	NS	NS	0.95**	0.97**	NS	NS	NS	NS	NS	NS
$T_H(\alpha)$	NS	NS	NS	NS	NS	NS	NS	NS	NS	0.83**	NS	NS	NS	NS
$T_H(\alpha,\beta)$	NS	NS	NS	NS	NS	NS	NS	NS	0.73*	0.88**	NS	NS	NS	NS
$T_V(\alpha)$	NS	NS	0.82**	NS	NS	NS	NS	NS	NS	NS	0.79*	0.83**	NS	NS
$T_V(\alpha,\beta)$	NS	NS	0.82**	NS	NS	NS	NS	NS	NS	NS	0.75*	0.79*	NS	NS
$T_F(\alpha)$	NS	NS	NS	NS	NS	NS	NS	NS	NS	NS	NS	-0.71*	0.87**	NS
$T_F(\alpha,\beta)$	NS	NS	NS	NS	NS	NS	NS	NS	NS	NS	NS	NS	0.90**	0.70*

Similarly to the first metric $T_s(\alpha,\beta)$ then relies on simple thresholds applied on the emergence, first in frequency then in time. Its expression is as follows:

$$T_s(\alpha,\beta) = \frac{1}{N_t}\sum_{t=1}^{N_t} \mathbb{1}\left[\frac{\sum_{f=1}^{N_f}\Delta_s(t,f)\mathbb{1}_{\Delta_s(t,f)>\alpha}}{\sum_{f=1}^{N_f}\mathbb{1}_{\Delta_s(t,f)>\alpha}} > \beta\right] \quad (4)$$

where N_f is the number of third-octave bands. Here the emergence threshold α is applied to each frequency band of the signal at a given time frame. To determine if the source is heard in the frame a second threshold β is then used on the mean emergence of the source on emergent bands. Again, optimal values for parameters $\alpha_{opt} = -6dB$ and $\beta_{opt} = -5dB$ are found via grid search on the experiment corpus as no other subset of scenes with both physical and perceptual data is available. This set of values is more plausible physically, as it indicates that a source is considered heard if its sound level is at most 5 dB lower than that of other sources overlapping in time and frequency.

Table 1 shows the Pearson's correlation coefficients between the computed indicators and assessed parameters at the sound scene level (n=9). The three globally computed sound levels L_{50}, L_{A50} and $L_{50,1kHz}$ represent well the perceived overall loudness of the scene and can be used directly for pleasantness prediction. Ground truth emergences also correlate with the evaluated sound level parameters for all sources but traffic. The perceived time of presence is however represented poorly by its corresponding ground truth estimation in common background sources: traffic, birds and human voices. Traffic, specifically, is almost always present throughout a scene in real life conditions. Its ground truth activity thus does not vary significantly across the considered corpus, although high variations in perceptual assessments indicate that it may not be heard at all time. The two proposed indicators successfully account for this effect for background sources while yielding similar correlations for horns and footsteps. Furthermore, these parameters are more discriminative for traffic and birds. This confirms the need of an emergence-based time of presence indicator to successfully represent heard sources in the scene's mix.

For all sources the perceived time of presence and sound level are highly correlated ($r > 0.8, p < 0.01$). This is not the case for the corresponding acoustic indicators, indicating information redundancy between these two quantities at the perceptual level. As a result one of the two quantities is often omitted in proposed pleasantness models.

4. CONCLUSION

A pilot experiment was performed to assess the relevance of predicting perceptual parameters from acoustic indicators in simulated scenes for soundscape quality assessment. The ground truth time of presence of sources is found not sufficient to fully characterize soundscape perception. Some sources can be active but not heard in the mix, especially background sounds such as traffic. This illustrates the need to design a masking model-based metric to determine each source's perceptual importance in complex soundscapes. The proposed indicator $T_s(\alpha,\beta)$, while relying on a basic emergence model due to the small amount of available data, can be directly linked to source-specific perceptual quantities. Predicting the average pleasantness of a soundscape can thus be achieved by estimating the source activity and emergence indicators proposed in Section 2.

Precision requirements of the postulated physical metrics are also obtained. 125 ms or longer time scales used for the computation of all indicators in the presented experiment allow the design of perceptually relevant indicators. A binary masking model is shown in this study to improve parameter prediction. The estimation of source-wise emergence as a classification process (e.g. 4 classes from *Not heard at all* to *Dominant*) as opposed to continuous regression is thus sufficient for the application needs.

Future work will 1) consider a refined perceptual experiment with a richer soundscape corpus in order to achieve a stronger validation and model design including comparison with state-of-the-art masking models and 2) formulate a complete experimental protocol dedicated to the soundscape characterization task.

5. REFERENCES

[1] EC, "Directive 2002/49/ec of the european parliament and of the council of 25 june 2002 relating to the assessment and management of environmental noise," *Off. J. Eur. Communities*, vol. 189, p. 12, 2002.

[2] C. Mydlarz, J. Salamon, and J. Bello, "The implementation of low-cost urban acoustic monitoring devices," *Applied Acoustics*, vol. 117, pp. 207–218, 2017.

[3] F. Gontier, M. Lagrange, P. Aumond, A. Can, and C. Lavandier, "An efficient audio coding scheme for quantitative and qualitative large scale acoustic monitoring using the sensor grid approach," *Sensors*, vol. 17, 2017.

[4] J. Picault, A. Can, J. Ardouin, P. Crepeaux, T. Dhome, D. Ecotiere, M. Lagrange, C. Lavandier, V. Mallet, C. Mietlicki, and M. Paboeuf, "Characterization of urban sound environments using a comprehensive approach combining open data, measurements, and modeling," in *Acoustics '17, Boston*, 2017.

[5] ISO 12913-1:2014, "Acoustics - soundscape - part 1: definition and conceptual framework," International Organization for Standardization, Geneva, CH, Standard, 2014.

[6] B. Berglund and M. Nilsson, "On a tool for measuring soundscape quality in urban residential areas," *Acta Acust. unit. Acust.*, vol. 92, pp. 938–944, 2006.

[7] A. Brown, "Towards standardization in soundscape preference assessment," *Applied Acoustics*, vol. 72, pp. 387–392, 2011.

[8] F. Aletta, J. Kang, and O. Axelsson, "Soundscape descriptors and a conceptual framework for developing predictive soundscape models," *Landsc. Urban Plan.*, vol. 149, pp. 65–74, 2016.

[9] O. Axelsson, M. Nilsson, and B. Berglund, "A principal components model of soundscape perception," *J. Ac. Soc. Am.*, vol. 128, p. 2836, 2010.

[10] R. Cain, P. Jennings, and J. Poxon, "The development and application of the emotional dimensions of a soundscape," *Applied Acoustics*, vol. 74, pp. 232–239, 2013.

[11] B. D. Coensel and D. Botteldooren, "The quiet rural soundscape and how to characterize it," *Acta Acust. unit. Acust.*, vol. 92, pp. 887–897, 2006.

[12] P. Delaitre, C. Lavandier, C. Ribeiro, M. Quoy, E. D'Hondt, E. G. Boix, and K. Kambona, "Influence of loudness of noise events on perceived sound quality in urban context," in *Inter Noise*, 2014.

[13] P. Ricciardi, P. Delaitre, C. Lavandier, F. Torchia, and P. Aumond, "Sound quality indicators for urban places in paris cross-validated by milan data," *J. Ac. Soc. Am.*, vol. 138, pp. 2337–2348, 2014.

[14] P. Aumond, A. Can, B. D. Coensel, D. Botteldooren, C. Ribeiro, and C. Lavandier, "Modeling soundscape pleasantness using perceptive assessments and acoustic measurements along paths in urban context," *Acta Acust. unit. Acust.*, vol. 103, pp. 430–443, 2017.

[15] C. Lavandier and B. Defreville, "The contribution of sound source characteristics in the assessment of urban soundscapes," *Acta Acust. unit. Acust.*, vol. 92, pp. 912–921, 2006.

[16] M. Nilsson and B. Berglund, "Soundscape quality in suburban green areas and city parks," *Acta Acust. unit. Acust.*, vol. 92, pp. 903–911, 2006.

[17] A. Can, L. Leclerq, J. Lelong, and J. Defrance, "Capturing urban traffic noise dynamics through relevant descriptors," *Applied Acoustics*, vol. 69, pp. 1270–1280, 2008.

[18] A. Can, P. Aumond, S. Michel, B. D. Coensel, C. Ribeiro, D. Botteldooren, and C. Lavandier, "Comparison of noise indicators in an urban context," in *45th International Congress and Exposition of Noise Control Engineering*, 2014.

[19] L. Brocolini, C. Lavandier, M. Quoy, and C. Ribeiro, "Measurement of acoustic environments for urban soundscapes: choice of homogeneous periods, optimization of durations, and selection of indicators," *J. Ac. Soc. Am.*, vol. 134, pp. 813–821, 2013.

[20] J. Gloaguen, A. Can, M. Lagrange, and J. Petiot, "Estimating traffic noise levels using acoustic monitoring: a preliminary study," in *DCASE 2016, Detection and Classification of Acoustic Scenes and Events*, 2016.

[21] J. P. Bello, C. Mydlarz, and J. Salamon, "Sound analysis in smart cities," in *Computational Analysis of Sound Scenes and Events*, T. Virtanen, M. D. Plumbley, and D. P. W. Ellis, Eds. Springer International Publishing, 2018, pp. 373–397.

[22] M. Nilsson, D. Botteldooren, and B. D. Coensel, "Acoustic indicators of soundscape quality and noise annoyance in outdoor urban areas," in *19th International Congress on Acoustics*, 2007.

[23] J. Gloaguen, A. Can, M. Lagrange, and J. Petiot, "Creation of a corpus of realistic urban sound scenes with controlled acoustic properties," in *Proceedings of Meetings on Acoustics*, 2017.

[24] G. Lafay, M. Lagrange, M. Rossignol, E. Benetos, and A. Roebel, "A morphological model for simulating acoustic scenes and its application to sound event detection," *IEEE/ACM Transactions on Audio, Speech and Language Processing*, vol. 24, no. 10, pp. 1854–1864, 2016. [Online]. Available: https://hal.archives-ouvertes.fr/hal-01111381

[25] J. Jeon, J. Hong, C. Lavandier, J. Lafon, O. Axelsson, and M. Hurtig, "A cross-national comparison in assessment of urban park soundscapes in france, korea, and sweden through laboratory experiments," *Applied Acoustics*, vol. 133, pp. 107–117, 2018.

[26] M. Berzborn, R. Bomhardt, J. Klein, J. Richter, and M. Vorlander, "The ita-toolbox: an open source matlab toolbox for acoustic measurements and signal processing," in *43th Annual German Congress on Acoustics*, 2017.

LARGE-SCALE WEAKLY LABELED SEMI-SUPERVISED SOUND EVENT DETECTION IN DOMESTIC ENVIRONMENTS

Romain Serizel[1], *Nicolas Turpault*[1*], *Hamid Eghbal-Zadeh*[2†], *Ankit Parag Shah*[3]

[1]Universit de Lorraine, CNRS, Inria, Loria, F-54000 Nancy, France
[2]Institute of Computational Perception, Johannes Kepler University of Linz, Austria
[3]Language Technologies Institute, Carnegie Mellon University, Pittsburgh PA, United States

ABSTRACT

This paper presents DCASE 2018 task 4. The task evaluates systems for the large-scale detection of sound events using weakly labeled data (without time boundaries). The target of the systems is to provide not only the event class but also the event time boundaries given that multiple events can be present in an audio recording. Another challenge of the task is to explore the possibility to exploit a large amount of unbalanced and unlabeled training data together with a small weakly labeled training set to improve system performance. The data are Youtube video excerpts from domestic context which have many applications such as ambient assisted living. The domain was chosen due to the scientific challenges (wide variety of sounds, time-localized events...) and potential applications.

Index Terms— Sound event detection, Large scale, Weakly labeled data, Semi-supervised learning

1. INTRODUCTION

We are constantly surrounded by sounds and we rely heavily on these sounds to obtain important information about what is happening around us. Ambient sound analysis aims at automatically extracting information from these sounds. It encompasses disciplines such as sound scene classification (in which context does this happen?), sound event detection and classification (SED) (what happens during this recording?) [1]. It has been attracting a continuously growing attention during the past years as it can have a great impact in many applications including smart cities, autonomous cars or ambient assisted living.

In this task, we focus on SED with time boundaries in domestic applications. The system then has to detect when an sound event occurs and what is the class of the event (as opposed to audio tagging where only the presence of a sound event is important regardless of when it happened). Current systems heavily rely on a supervised training phase and usually require a large set of sound recordings labeled in terms of event with time boundaries. Obtaining such annotations is tedious and it is hardly feasible to gather a sufficient amount of data to train state-of-the-art systems that are often relying on complex deep network architectures [2, 3, 4, 5, 6].

We propose to follow-up on DCASE2017 task 4 [7] and investigate the scenario where a large scale corpus is available but only a small amount of the data is labeled. We propose to use a subset of the Audioset corpus [8] targeting classes of sound events related to domestic applications. The labels are provided at clip level (an event is present or not within a sound clip) but without the time boundaries (weak labels, that can also be refereed to as tags) in order to further decrease the annotation time. These constraints indeed correspond to constraints faced in many real applications where the budget allocated to annotation is limited.

In order to fully exploit this dataset, the proposed systems will have to tackle two different problems. The first problem is related to the exploitation of the unlabeled part of the dataset either in unsupervised approaches [9, 10] or together with the labeled subset in semi-supervised approaches [11, 12, 13]. The second problem is related to the detection of the time boundaries and how to train a system that can detect these boundaries from weakly labeled data [14, 15]. Currently, most of the state-of-the-art approaches rely mainly on smoothing techniques to ensure time consistency [6, 16, 17] which is not sufficient to estimate the time boundaries accurately. The evaluation metric chosen is penalizing these boundary estimation errors heavily in order to emphasize this latter aspect.

This manuscript describes DCASE2018 task 4 and is organized as follows. Section 2 presents the dataset used in task 4. The task evaluation procedure is described in Section 3. The baseline system is described and evaluated in Section 4. Conclusions and perspectives are presented in Section 5

2. AUDIO DATASET

The task employs a subset of Audioset [8]. Audioset consists of an expanding ontology of 632 classes of sound events and a collection of 2 million human-labeled 10-seconds sound clips (less than 21% are shorter than 10 seconds) drawn from 2 million YouTube videos. The ontology is specified as a hierarchical graph of event categories, covering a wide range of human and animal sounds, musical instruments and genres, and common everyday environmental sounds.

Audioset provides labels at clip level (without time boundaries for the events) also known as weak labels. Audio clips are collected from Youtube videos uploaded by independent users so the number of clips per class varies dramatically and the dataset is not balanced[1]. Google researchers conducted a quality assessment task where experts were exposed to 10 randomly selected clips for each class of sound events. In most of the cases not all the clips contains the event related to the given label. More information about the initial annotation process can be found in Gemmeke et al. [8].

*This work has been funded by the French region Grand-Est.

†This work has been funded by the Austrian Ministries BMVIT and BMWFW, and the Province of Upper Austria, via the COMET Center SCCH.

[1]see also `https://research.google.com/Audioset//dataset/index.html`

Class	Count			Event duration (in s)	
	Training	Test		Test	
	clips	clips	events	mean	median
Alarm/bell/ringing	205	45	112	1.53	0.58
Blender	134	30	40	5.35	4.59
Cat	173	32	97	0.81	0.71
Dishes	184	35	122	0.56	0.42
Dog	214	29	127	1.03	0.66
Electric shaver/toothbrush	103	25	28	7.41	8.78
Frying	171	24	24	9.34	10.00
Running water	343	63	76	5.61	5.53
Speech	550	105	261	1.51	1.20
Vacuum cleaner	167	35	36	8.66	9.99
Total	2244	288	906	2.41	1.03

Table 1: Class-wise statistics

Number of classes	1	2	3 and more
Clip proportion	62.36%	32.89%	4.75%

Table 2: Proportion of clips with multiple classes of sound events (training set)

We will focus on a subset of Audioset that consists of 10 classes of sound events (Table 1). The development set provided for task 4 is split into a training set and a test set[2].

2.1. Training set

In order to reflect what could possibly happen in a real-world scenario, we provide 3 different splits of training data in task 4 training set: a labeled training set, an unlabeled in domain training set and an unlabeled out of domain training set (clips that do not contains the classes listed in Table 1, see also below).

2.1.1. Labeled training set

This set contains 1,578 clips (2,244 class occurrences) for which weak labels provided in Audioset have been verified and corrected by human annotators (classes that were not present in the audio clip were removed and missing classes were addedp). The weak labels are provided in a tab-separated csv file under the following format:

[filename (str)][tab][class_label (str)]

The first column is the name of the sound clip downloaded from YouTube composed of the YouTube ID of the video and the time boundaries of the 10-seconds audio clip extracted from the video. The last column corresponds to the sound events that are present in the clip, each separated by a semi-colon.

The amount of clips per class of sound events is presented in Table 1 and the number of classes observed per clip is presented in Table 2. One-third of the clips in this set contain at least two different classes of sound events.

2.1.2. Unlabeled in domain training set

This set contains 14,412 clips. The clips are selected such that the distribution per class of sound event (based on Audioset labels) is close to the distribution in the labeled set. Note however that given the uncertainty on Audioset labels this distribution might not be exactly similar. Audioset labels have not been verified for this subset and should not be used during the systems development.

[2]The annotations files and the script to download the audio files is available on the git repository for task 4 `https://github.com/DCASE-REPO/dcase2018_baseline/tree/master/task4/dataset`

Number of events	1	2	3 and more
Time proportion	86.86%	10.40%	2.74%
Clip proportion	33.68%	17.71%	48.61%

Table 3: Proportion of overlapping events (test set)

2.1.3. Unlabeled out of domain training set

This set is composed of 39,999 clips extracted from classes of sound events that are not considered in the task. Note that these clips are chosen based on the original Audioset labels that might be noisy. Additionally, as the speech class is present in about half of the clips in Audioset, the unlabeled out of domain set also contains almost 20000 clips with speech. This was the only way to have a set which is somehow representative of Audioset. Indeed, discarding speech would have also meant discarding many other classes of sound events and the variability of the set would have been penalized. Audioset labels have not been verified for this subset and should not be used during the systems development.

2.2. Test set

The test set is designed such that the distribution in term of clips per class of sound event is similar to that of the weakly labeled training set. The size of the test set is such that it represents about 20% of the size of the weakly labeled training set, it contains 288 clips (906 events). The test set is annotated with strong labels, with time boundaries (obtained by human annotators).

The minimum length for an event is 250 ms. The minimum duration of the pause between two events from the same class is 150 ms. When the silence between two consecutive events from the same class was less than 150 ms the events have been merged to a single event. The strong labels are provided in a tab-separated csv file under the following format:

[filename (str)][tab][onset (float)][tab][offset (float)][tab][class_label (str)]

The first column, is the name of the audio file downloaded from YouTube, the second column is the onset time in seconds, the third column is the offset time in seconds and the last column corresponds to the class of the sound event.

The amount of events per class is presented in Table 1. This table also presents the mean and median duration of the events for each class. From these durations it is possible to categorize the events into three classes: short events (Alarm, cat, dishes, dog and speech), events with variable duration (blender and running water) and long events that span almost over the whole clip (electric shaver, frying and vacuum cleaner). This classification is confirmed by the observation of the duration distribution presented on Figure 1.

One of the focus of this task is the development of approaches that can provide fine time-level segmentation while learning on weakly labeled data. The observation of the event duration distribution confirms that in order to perform well it is essential to design approaches that are efficient at detecting both short events and events that have a longer duration.

Table 3 presents the time proportion of overlapping events in the test set. With overlapping events occurring about 12% of the

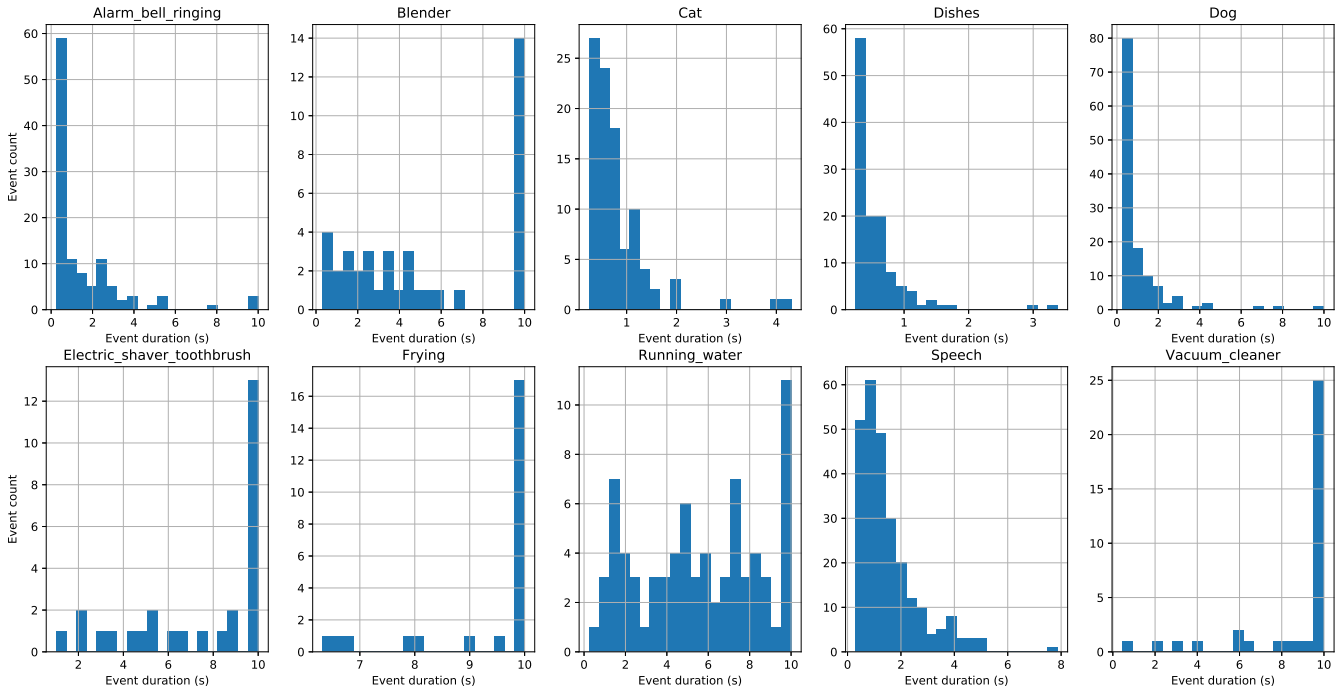


Figure 1: Duration distribution by class of sound events.

time it is important to design approaches that are able to detect and properly classify the overlapping events. However, this proportion is lower than the proportion of the clips containing multiple sound events (about 64%). This tends to confirm that being able to discriminate between events that occur at different times within the clip is of a high importance as well. This aspect reinforces the focus on efficient time segmentation.

2.3. Evaluation set

The evaluation set contains 880 10-seconds audio clips. The process to select the clips was similar to the process applied to select clips in the training set and the test set, in order to obtain a set with comparable classes distribution. Labels with time boundaries (obtained by human annotators) will be released after the DCASE 2018 challenge is concluded.

3. TASK DESCRIPTION

The task consists of detecting sound events within web videos using weakly labeled training data. The detection within a 10-seconds clip should be performed with start and end timestamps. Task rules are detailed on the challenge webpage[3].

3.1. Task evaluation

Submissions will be evaluated with event-based measures for which the system output is compared to the reference labels event by event [18]. True positives are the occurrences when an event present in the system output corresponds to an event in the reference annotations. The correspondence between event boundaries are estimated with a 200ms collar tolerance on onsets and a tolerance on offsets that is the maximum of 200ms and 20% of the event length. False positives are obtained when an event is present in the system output but not in the reference annotations (or not within the tolerance on the onset or the offset). False negatives are obtained when an event is present in the reference annotations but not in the system output (or not within the tolerance).

Submissions will be ranked according to the event-based F1-score. The F1-score is first computed class-wise over the whole evaluation set:

$$F1_c = \frac{2TP_c}{2TP_c + FP_c + FN_c}, \quad (1)$$

where TP_c, FP_c and FN_c are the number of true positives, false positives and false negative for class of sound event c over the whole evaluation set, respectively.

The final score is the F1-score average over class regardless of the number of events per class (macro-average):

$$F1_{\text{macro}} = \frac{\sum_{c \in \mathcal{C}} F1_c}{n_{\mathcal{C}}}, \quad (2)$$

where $\mathcal{C}$ is the classes ensemble and $n_{\mathcal{C}}$ the number of classes.

Evaluation is done using sed_eval toolbox [18]. The choice of the macro averaging allows for according the same importance to each class of sound event in an unbalanced scenario as in task 4.

4. BASELINE

4.1. System description

The baseline system is based on convolutional recurrent neural networks (CRNN). The signals are sampled at 44.1 kHz and 64 log

[3] http://dcase.community/challenge2018/

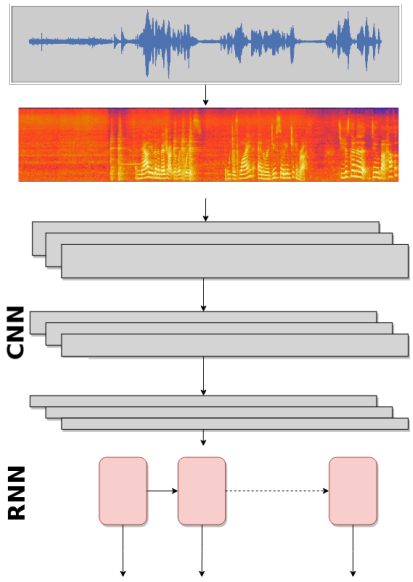


Figure 2: Baseline system description

mel-band magnitudes are extracted from 40 ms frames with 50% overlap. Using these features, we train a first CRNN with three convolution layers (64 filters (3x3), max pooling (4) along the frequency axis and 30% dropout), one recurrent layer (64 Gated Recurrent Units with 30% dropout on the input), a dense layer (10 units sigmoid activation) and global average pooling across frames (Figure 2).

The system is trained for 100 epochs (early stopping after 15 epochs patience) on weak labels ('train/weak', 1,578 clips). 20% of this set is used for validation. This model is trained at clip level, the annotations only indicate if the event is present or not during the clip. The inputs are 500 frames long for a single output frame. This first model is used to predict labels of unlabeled files ('train/unlabel_in_domain', 14,412 clips).

A second model based on the same architecture (Figure 2) is trained on predictions of the first model on the unlabeled subset ('train/unlabel_in_domain'). The files with manual annotations ('train/weak') are used for the validation of the model. The main difference with the first pass model is that the output dense layer is time-distributed in order to be able to predict events at the frame level. The inputs are 500 frames long, each of them labeled identically following clip labels. The model outputs a decision for each of these 500 frames. Median filtering over 51 frames ($\approx$ 1 s) is applied to the output of the network, in order to obtain the events onset and offset for each file. The full process is described in Algorithm 1.

4.2. System performance

The F1-score performance of the baseline system after the first pass (training on 'train/weak' data) and after the second pass (training on 'train/unlabel_in_domain' with labels obtained with the first pass system) are presented in Table 4. The second pass improves performance compared to the first pass in macro-average and for all classes expect for 'Dishes'. Therefore, the baseline system takes advantage of the unlabeled data even though this could probably be done more efficiently by the systems submitted to the task.

The F1-score is close to zero for all the short event classes. It is slightly higher for event with variable duration (running water and blender) and performs the best for the long event class. This tends to indicate that the baseline performs very poorly in time-segmentation which is one of the focus of this task. Errors in segmentation are heavily penalized by the metric as they result in both a false positive and a false negative. Therefore, in order to improve performance, submitted systems should propose efficient time-segmentation approaches.

Algorithm 1 Script description

procedure DOWNLOAD THE DATA (ONLY THE FIRST TIME)
end procedure
procedure FIRST PASS (at clip level)
    Train a CRNN on weakly labeled data ('train/weak')
        ▷ 20% of data used for validation
    Predict class for unlabeled data ('train/unlabel_in_domain')
end procedure
procedure SECOND PASS (at frame level)
    Train a CRNN on labels predicted for the unlabeled data during the first pass ('train/unlabel_in_domain')
        ▷ weak data ('train/weak') is used for validation
        ▷ Labels are used at frames level. If a class is present in the clip, all frames contain the label.
    Predict strong test labels ('test/')
        ▷ Predict an event with an onset and an offset
end procedure
procedure EVALUATION (event based)
    Evaluate the prediction on the test set with respect to manual annotations
end procedure

Class	1^{st} pass	2^{nd} pass
Alarm/bell/ringing	3.2%	3.9%
Blender	10.1%	15.4%
Cat	0.0%	0.0%
Dishes	1.9%	0.0%
Dog	0.0%	0.0%
Electric shaver/toothbrush	18.2%	32.4%
Frying	9.4%	31.0%
Running water	7.6%	11.4%
Speech	0.0%	0.0%
Vacuum cleaner	24.8%	46.5%
Macro average	**7.51%**	**14.06%**

Table 4: Event based F1-scores

5. CONCLUSION

This paper presents DCASE 2018 task 4 on large-scale weakly labeled semi-supervised SED in domestic environments. The goal is to exploit a small dataset of weakly labeled sound clips (without time boundaries) together with a larger unlabeled dataset to perform SED (with time boundaries). This indeed corresponds to a realistic scenario as obtaining time coded annotations is time consuming and is generally considered as one of the principal bottlenecks in training SED systems. The design of the dataset with a wide variability in event lengths and the choice of a metric that is heavily penalizing segmentation errors put a strong focus on the problem of localizing the events in time.

6. REFERENCES

[1] T. Virtanen, M. D. Plumbley, and D. Ellis, *Computational analysis of sound scenes and events*. Springer, 2018.

[2] G. Parascandolo, H. Huttunen, and T. Virtanen, "Recurrent neural networks for polyphonic sound event detection in real life recordings," in *Proc. ICASSP*, Mar. 2016, pp. 6440–6444.

[3] N. Takahashi, M. Gygli, B. Pfister, and L. Van Gool, "Deep convolutional neural networks and data augmentation for acoustic event detection," in *Proc. INTERSPEECH*, Apr. 2016.

[4] S. Adavanne, G. Parascandolo, P. Pertilä, T. Heittola, and T. Virtanen, "Sound event detection in multichannel audio using spatial and harmonic features," in *arXiv preprint arXiv:1706.02293*, 2017.

[5] E. Cakir, G. Parascandolo, T. Heittola, H. Huttunen, and T. Virtanen, "Convolutional recurrent neural networks for polyphonic sound event detection," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 25, no. 6, pp. 1291–1303, 2017.

[6] Y. Xu, Q. Kong, W. Wang, and M. D. Plumbley, "Large-scale weakly supervised audio classification using gated convolutional neural network," in *Proc. DCASE*, 2017.

[7] A. Mesaros, T. Heittola, A. Diment, B. Elizalde, A. Shah, E. Vincent, B. Raj, and T. Virtanen, "Dcase 2017 challenge setup: Tasks, datasets and baseline system," in *Proc. DCASE*, 2017.

[8] J. F. Gemmeke, D. P. W. Ellis, D. Freedman, A. Jansen, W. Lawrence, R. C. Moore, M. Plakal, and M. Ritter, "Audio set: An ontology and human-labeled dataset for audio events," in *Proc. ICASSP*, 2017.

[9] J. Salamon and J. P. Bello, "Unsupervised feature learning for urban sound classification," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, April 2015, pp. 171–175.

[10] A. Jansen, M. Plakal, R. Pandya, D. Ellis, S. Hershey, J. Liu, C. Moore, and R. A. Saurous, "Unsupervised learning of semantic audio representations," in *Proc. ICASSP*, 2018.

[11] Z. Zhang and B. Schuller, "Semi-supervised learning helps in sound event classification," in *Proc. ICASSP*, 2012, pp. 333–336.

[12] T. Komatsu, T. Toizumi, R. Kondo, and Y. Senda, "Acoustic event detection method using semi-supervised non-negative matrix factorization with a mixture of local dictionaries," in *Proc. DCASE)*, 2016, pp. 45–49.

[13] B. Elizalde, A. Shah, S. Dalmia, M. H. Lee, R. Badlani, A. Kumar, B. Raj, and I. Lane, "An approach for self-training audio event detectors using web data," in *Proc. EUSIPCO*, 2017, pp. 1863–1867.

[14] A. Kumar and B. Raj, "Audio event detection using weakly labeled data," *CoRR*, vol. abs/1605.02401, 2016.

[15] ——, "Audio event and scene recognition: A unified approach using strongly and weakly labeled data," in *Proc. IJCNN*. IEEE, 2017, pp. 3475–3482.

[16] I.-Y. Jeong, S. Lee, Y. Han, and K. Lee, "Audio event detection using multiple-input convolutional neural network," in *Proc. DCASE*, 2017, pp. 51–54.

[17] J. Lee, J. Park, S. Kum, Y. Jeong, and J. Nam, "Combining multi-scale features using sample-level deep convolutional neural networks for weakly supervised sound event detection," in *Proc. DCASE*, 2017, pp. 69–73.

[18] A. Mesaros, T. Heittola, and T. Virtanen, "Metrics for polyphonic sound event detection," *Applied Sciences*, vol. 6, no. 6, p. 162, May 2016.

THE AALTO SYSTEM BASED ON FINE-TUNED AUDIOSET FEATURES FOR DCASE 2018 TASK2 —— GENERAL PURPOSE AUDIO TAGGING

Zhicun Xu, Peter Smit, Mikko Kurimo

Aalto University, Department of Signal Processing and Acoustics, Espoo, Finland,
{zhicun.xu, peter.smit, mikko.kurimo}@aalto.fi

ABSTRACT

In this paper, we presented a neural network system for DCASE 2018 task 2, general purpose audio tagging. We fine-tuned the Google AudioSet feature generation model with different settings for the given 41 classes on top of a fully connected layer with 100 units. Then we used the fine-tuned models to generate 128 dimensional features for each 0.960s audio. We tried different neural network structures including LSTM and multi-level attention models. In our experiments, the multi-level attention model has shown its superiority over others. Truncating the silence parts, repeating and splitting the audio into the fixed length, pitch shifting augmentation, and mixup techniques are all used in our experiments. The proposed system achieved a result with MAP@3 score at 0.936, which outperforms the baseline result of 0.704 and achieves top 8% in the public leaderboard.

Index Terms— audio tagging, AudioSet, multi-level attention model

1. INTRODUCTION

Sound contains various information that could indicate the sound sources, surrounding environment, music genres, possible dangers or even the emotions of the speakers. Thus sound plays a crucial part in our daily communication and interaction with the world. Teaching machines to listen, such as recognizing the sound events, would benefit humans in many ways. Relevant applications include public surveillance, sound print, auditory medical information monitoring and multimedia content analysis.

General purpose audio tagging is a task that infers descriptive labeling from these sounds such as musical instruments, domestic animals, and human activities. Recognizing and labelling these sound events with appropriate tags can provide a powerful tool to categorize the extensively large amount of audio data from the internet. With the labels, the content providers can give better services such as providing audio descriptions for visually or hearing impaired people, and providing powerful searching tools for the people working in the entertainment industries.

The traditional methods for doing the audio classification and audio tagging are adapted from speech recognition such as the Mel-frequency cepstral coefficients (MFCC) features and simple Gaussian mixture model (GMM) classifiers [1]. Recently, deep neural networks have proven its great usefulness in feature engineering, classification, detection, and audio synthesis. Almost all the submissions in DCASE 2017 [2] used some forms of neural networks such as long short-term memory units (LSTM) and convolutional neural networks (CNN). Thus deep learning is our main research approach for this task. Google has created a dataset called AudioSet with a structured hierarchical ontology [3], which provides the proper way to annotate the sounds. Instead of releasing the original audio files, each sample from the AudioSet is represented by 10 instances of 128 dimensional features. Google also provides a pre-trained model to generate the 128 features for 0.960 seconds audio. Kong *et al*. proposed a single-level attention model on this dataset, which outperformed the Google' baseline [4]. Later, Yu *et al*. proposed a multi-level attention model as an extension to previous single-level attention models, the results outperformed both the single-level attention model and the baseline [5].

Thus we came up with the idea to fine-tune the feature generation model first for the 41 classes of this DCASE task, and then try the multi-level attention model on the generated compact features. We aimed at proving the usability of these compact features and the superiority of multi-level attention model. We also incorporated pitch shifting augmentation and mixup techniques.

The structure of this paper is as follows. Section 2 describes methods on how to fine-tune the existing CNN model for the given 41 classes. Section 3 describes the design of our proposed system. The experimental results and conclusions can be seen in section 4 and 5 respectively.

2. FINE-TUNED VGGISH MODEL

2.1. Structure

VGGNet [6] with deep CNN structures has worked greatly well in image classification. Since the spectral representation of an audio signal can be used directly as an image, this deep CNN structure is also a promising techniques in many machine listening tasks, such as audio tagging, audio event detection and acoustics scene classification. VGGish model [7] is a variant of the VGG model with minor modifications. Table 1 shows the detailed structure of the modified version. Google trained this model on the YouTube-100M dataset with a total of 100 million videos. The training set contains 70 million videos and they were further split into non-overlapping 960 ms audio frames. Log-mel spectrograms were then computed as 96×64 images for the input of the VGGish model. The evaluation results of VGGish model showed its great usability in the audio domain.

The pre-trained VGGish model can act as a feature extractor. The provided 128 embedding features for 0.960 seconds are very compact, high level and semantically meaningful as well. These

This work was supported by the Kone foundation, and the European Union's Horizon 2020 research and innovation programme via the project MeMAD (GA780069). Computational resources were provided by the Aalto Science-IT project.

Table 1: VGGish model structure. The kernel size for CNN is (3, 3) and the kernel size for pooling is (2, 2). The activation function is 'relu' for all the layers.

layers	filters@size	layers	filters@size
1. input	1@96×64	**9. conv**	512@12×8
2. conv	64@96×64	**10. conv**	512@12×8
3. pooling	64@48×32	**11. pooling**	512@6×4
4. conv	128@48×32	**12. flatten**	12288
5. pooling	128@24×16	**13. fc**	4096
6. conv	256@24×16	**14. fc**	4096
7. conv	256@24×16	**15. fc**	128
8. pooling	256@12×8	**16. ...**	...

compact features can then be fed into a shallower model for classification. VGGish model can also become part of a larger model where more layers are added upon the model, which makes it possible to adapt and fine-tune this model for different datasets.

2.2. Balanced and Unbalanced Fine-tuning

The original VGGish model is trained for a multi-label classification task with the Google AudioSet ontology [3], which has 632 classes in a hierarchy tree structure. This challenge is a multi-class classification task, which means there is only one correct label for a sample. The 41 classes for this task all belongs to the AudioSet ontology. The training data is provided by FreeSound Dataset (FSD) [8]. The dataset has in total 9473 training files with the smallest class having only 94 training samples and the biggest class having 300 training samples. The average length of the audio files is 6.7 seconds. The more detailed description of the task setup, dataset and baseline can be seen in [9]. We added one hidden layer with 100 units after the VGGish model, and the final classification layer has 41 units with softmax activations.

To fine-tune the VGGish model, we must divide the data into a training part and a validation part. The validation loss is then used as the stopping criterion in case of overfitting. Firstly, we used the first 8000 audio files as training data and the remaining 1473 audio files as validation data. To fully use the data, we then used the last 8000 audio files as training data and the first 1473 audio files as validation data. Thus for one balancing technique we would get two models, one of which is fine-tuned on the first 8000 audio files and the other is fine-tuned on the last 8000 audio files.

Mini-batch balancing is a technique which assigns equal number of training samples for each class in each mini-batch. It has been proven useful on AudioSet to deal with the unbalanced dataset [4]. For the balanced fine-tuning, we followed the mini-batch balancing method by choosing an audio sample from each class for every training batch. However, it is not perfect balancing since audio samples may contain different number of 0.960 seconds length segments. In total, there would be 41 audio files, around 280 log-mel spectrograms, for each training batch. On the other hand, we also fine-tuned a unbalanced model by randomly choose 41 audio files in each batch for comparison. So we got 4 models in total.

The fine-tuned model are used as the feature extractors for the audio. Each audio file is firstly split into several non-overlapping 0.960 seconds segments. And for each segment, the log-mel spectrogram with dimension 96×64 is computed, then the spectrogram is fed into the fine-tuned model to get the 128 dimensional features.

3. THE PROPOSED SYSTEM

3.1. Preprocessing

The provided audio files are provided as PCM 16 bits, 44.1 kHz, mono format. However, the original quality might be quite different since they are uploaded by users all around the world. For preprocessing, we trimmed the silence parts only in the beginning and the end to avoid the influence of these irrelevant silence parts. We used the librosa[1] toolbox for the trimming here. The normalizing and pitch shifting mentioned in the following section are also implemented using this toolbox. The silence parts that are in the middle might contain important dynamic information for classification. We then normalized the amplitude of the audio files to [-1, 1]. The trimmed and normalized audio files are then split into non-overlapping 0.960 seconds segments. For each segment the log-mel spectrogram with size 96×64 is computed with 25ms window size and 10ms hop size. The 96 represent the frame size and 64 represents the number of frequency bands. The spectrograms are then fed into the fine-tuned VGGish model for features extraction.

3.2. Data Augmentation

The data is unbalanced where only around half of the classes have 300 audios. The imbalance problem could make the model emphasize more on the classes with more training samples and neglect to learn from the classes with less samples. To deal with this problem, we used the pitch shifting, repeat and split strategies for augmentation.

For the implementations of pitch shifting, we randomly choose an integer number between (-12, 12) for each audio file, and then shift the correspond number of semitones to create the new file. The shifting does not affect much on the melodic or stable classes, since the maximum shift is only one octave. For the noise-like classes such as 'fireworks' and 'fart', the perception is still relatively okay even for those with maximum amount of shift.

Some of our models used fixed length audio as input, but the audio files have variable length. To fully use the full length of the audio, we split them into several fixed length audios as input, which also gave us more training data. For the audio files that are shorter, we repeat them and concatenate them together to the fixed length.

3.3. Mixup

Mixup means training the neural networks using the convex combinations from pairs of examples and their labels. It helps the neural network to emphasize the linear combination between the training samples, which can improve the generalization ability, reduce memorization of the corrupted labels, increase the robustness to adversarial examples [10]. The training data provided by the organizer has another label representing if the audio is manually verified. Thus trying this method can help reduce the effect of the potential corrupted or misclassified audio in the unverified audios. Besides, this will also allow the model learn to distinguish between classes. The following equation [10] explains how the algorithms works:

$$\begin{aligned} \tilde{x} &= \lambda x_i + (1-\lambda) x_j \\ \tilde{y} &= \lambda y_i + (1-\lambda) y_j \\ \lambda &\sim \text{Beta}(\alpha, \alpha) \\ \alpha &\in (0, \infty) \end{aligned} \tag{1}$$

[1] https://github.com/librosa/librosa

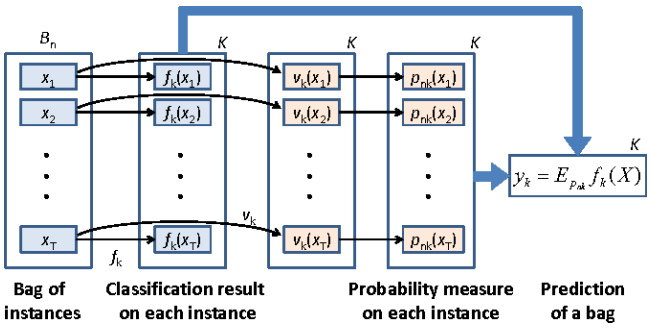


Figure 1: Attention model for Audio Set [4]

The (x_i, y_i) and (x_j, y_j) are two training and label pairs, which are randomly sampled form the training data. λ is drawn from a Beta distribution and lies in the region [0,1]. The experiments in the work [10] shows that increasing the α would increase the training error on real data and minimize the generalization gap. They also found that $\alpha \in [0.1, 0.4]$ would give an improved performance and large α will leads to underfitting.

3.4. Multi-level Attention Model

The dynamic changes are crucial in audio tagging challenges. Only considering the frequency structure might be useful for the recognition of the melodic instruments, but for the tags such as 'gunshort', and 'knock', the temporal changes are much more important. An attention model, which assign different weights for the instances in a time series, is a good strategy for considering the dynamics of sound. An attention model structure [4] presented for the AudioSet can be seen in Fig. 1. B_n represents an audio file with 10 instances $x_1 \sim x_{10}$, $f_k(x_n)$ represents the classification results for class K. The weights for each instance are firstly computed as $v_k(x_1) \sim v_k(x_{10})$ and then normalized to $p_{nk}(x_1) \sim p_{nk}(x_{10})$ so that $\sum_i^{10} p_{nk}(x_i) = 1$. Finally, multiplying the weights with the relevant prediction results and summing them together gives the final prediction for class K.

Combining the features from different levels and different timescale can provide more accurate descriptions. A CNN-based architecture [11] has shown great performance for music tagging by aggregating the multi-level and multi-scale features. Concatenated features extracted from different levels of CNN has also been proven useful in computer vision tasks [12]. Inspired by the works [4] [5] on Google Audio Set, we decided to choose a similar multi-level attention structure for audio. We did not try multi-scale methods because the fine-tuned VGGish models can only generate features for the 0.960 seconds fixed length audio. We prepared each training sample as 6 concatenated segments, each segment contains 0.960 seconds audio. Then the 6×128 dimensional features extracted using the fine-tuned VGGish model are fed into the neural network for training.

Fig. 2 shows the model structure. Each of the 6 instances are fed into a fully connected neural networks with 3 layers. Different color boxes represent the different levels features, and the features from the same level are then fed into an attention model, shown in Fig. 1, to get the predictions. We then concatenated all three predictions from different levels and forward them into the final classification layer with the softmax activation to get the 41 probabilities for the 41 classes.

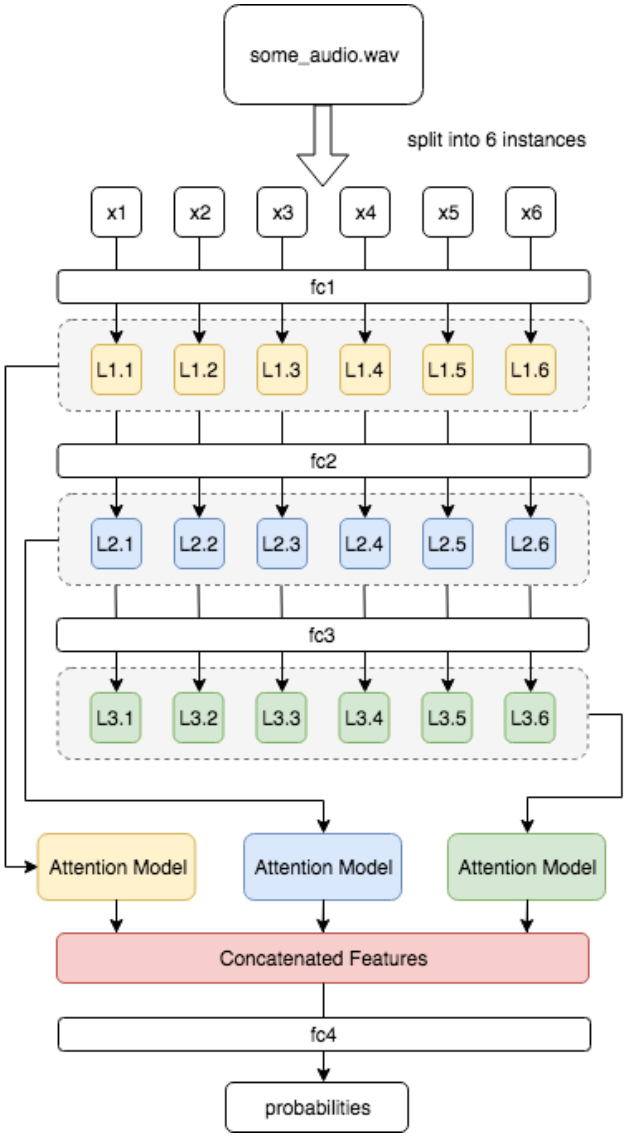


Figure 2: Model Structure for Multi-level Attention model

3.5. Evaluation Metric

The evaluation uses the Mean Average Precision @ 3 (MAP@3)[2]. The detailed implementation and explanation can be seen in the link provided in the footnote. Simply speaking, up to three predictions can be given even though there is only one correct label. The order of the predictions matters in this setting. If the correct label is predicted in the 1st position among those three predictions, the system would get a score 1. 2nd position would get a score 1/2, 3rd position would get a score 1/3. If there is no correct answer in the three predictions, the score would be zero. The average of the scores for all the testing samples would be the final evaluation score. However, the public leaderboard only has the score for around 19% of the testing data. So the scores in this paper are all evaluated upon

[2] https://github.com/benhamner/Metrics/blob/master/Python/ml_metrics/average_precision.py

Table 2: The one second DNN evaluation results on different fine-tuned VGGish models

Feature Extractor	MAP@3
Baseline	0.704
Original VGGish	0.606
Balanced First Part	0.870
Balanced Last Part	0.864
Balanced All	0.891
Unbalanced First Part	0.858
Unbalanced Last Part	0.872
Unbalanced All	0.892
Ensemble All	**0.903**

Table 3: Evaluation results on different model structures and hyper-parameters.

Index	Model Structure	MAP@3
A	1-Segment Multi-level	0.903
B	LSTM	0.914
C	6-Segment Multi-level Attention	0.925
D	C + Pitch shifting Augmentation	0.930
E	D + Mixup ($\alpha = 0.1$)	0.930
F	D + Mixup ($\alpha = 0.2$)	**0.936**
G	D + Mixup ($\alpha = 0.4$)	0.931
H	D + Mixup ($\alpha = 1.0$)	0.930

those 19% and the final results might be different.

4. EXPERIMENTS

4.1. Different fine-tuned VGGish models

Firstly, we tested the performance of different fine-tuned VGGish models on a simpler model structure. In this experiment, we built a simple 3-layer fully connected neural networks with each layer containing 600 units. The classification results from each layer are concatenated together as input to the final layer. The activation function is 'relu' for all the layers, except that the classification layers use 'softmax' activation function. Batch normalization and dropout with ratio 0.4 are used after each middle layer. The input is the 128 dimensional features for 0.960 seconds. For evaluation on files with different length, we took the average results as the score. As can be seen from Table 2, all of the fine-tuned models outperform the original VGGish model. 'Balanced first part' means the feature extractor is trained using mini-batch balancing on the first 8000 training samples and validated on the remaining ones. 'Balanced last part' means the feature extractor is trained using mini-batch balancing on the last 8000 training samples and validated on the remaining ones. 'Balanced all' means taking the geometric means of the results from the 'Balanced first part' and 'Balanced last part'. 'Unbalanced' means the model is trained without mini-batch balancing. The remaining names can be interpreted in similar way. There is no clear difference between the models trained with mini-batch balancing and those without. However, using geometric mean results of all 4 models will give the best results.

Table 4: Evaluation results on different number of segments multi-level attention models

Number of Segments	1	2	4	6
MAP@3	0.917	0.919	0.931	0.936
Number of Segments	8	10	Ensemble All	
MAP@3	0.929	0.931	0.931	

4.2. Different neural network structures and hyper parameters

Section 4.1 has shown that utilizing the results from all 4 fine-tuned models would give the best results. Thus for each model structure in this section, we also used the same strategy. The results for different model structures and different random mix factors α can be seen in Table 3. Model A has the same setting as the best results from Table 2. Model B uses one LSTM layer with 600 hidden units upon the original variable length input. Model C has the same 6-segment multi-level attention structures shown in Section 3.4, other parameter are same as A. Model D is based on model C with pitch shifting augmentation. Model E, F, G, H are based on D with different mixup factors.

From the comparison between model A, B, C, LSTM is better than the 1-segment multi-level model. The 6-segment multi-level attention model performs the best. Thus temporal information plays an important role in recognition, and multi-level attention model has better ability to model the temporal information than LSTM in our settings. Using the pitch shifting augmentation to generate a relevantly more balanced training data also improves the results a little bit. The comparison between model E, F, G, H shows mixup with the $\alpha = 0.2$ has the best performance among these 4 choices. However, since the difference among C-H is quite small, further significance test might still be needed.

4.3. Different number of segments

We also tried the multi-level attention model with different number of segments as input. Other training settings, such as the pitch shifting augmentation and mixup, are the same as the best results in Table 3. Results can be seen in Table 4. For the ensemble results, we took the geometric mean of all the results. We can see that the 6-segment model has the best performance and the ensemble does not improve the overall score. The reason might be that these models are not diverse enough.

5. CONCLUSIONS

In this paper, we tried different fine-tuning methods on the AudioSet VGGish model for generating 128 features for 0.960s audio. The results show that the combination of the 4 models trained with different train-validation splitting and balanced/unbalanced techniques would give the best results. We also implemented different neural network structures for comparison and found that multi-level attention model performs the best among all. This shows the importance of modeling the temporal information. The 6-segment multi-level attention model with pitch shifting augmentation and mixup method using $\alpha = 0.2$ has the best MAP@3 at 0.936 in public leaderboard. Further research might include more thorough fine-tuning, building own CNN model for feature generation, utilizing multi-scale features along with multi-level features for the attention model.

6. REFERENCES

[1] H. Aronowitz, "Segmental modeling for audio segmentation," in *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on*, vol. 4. IEEE, 2007, pp. IV–393.

[2] A. Mesaros, T. Heittola, A. Diment, B. Elizalde, A. Shah, E. Vincent, B. Raj, and T. Virtanen, "Dcase 2017 challenge setup: Tasks, datasets and baseline system," in *DCASE 2017-Workshop on Detection and Classification of Acoustic Scenes and Events*, 2017.

[3] J. F. Gemmeke, D. P. Ellis, D. Freedman, A. Jansen, W. Lawrence, R. C. Moore, M. Plakal, and M. Ritter, "Audio set: An ontology and human-labeled dataset for audio events," in *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*. IEEE, 2017, pp. 776–780.

[4] Q. Kong, Y. Xu, W. Wang, and M. D. Plumbley, "Audio set classification with attention model: A probabilistic perspective," *arXiv preprint arXiv:1711.00927*, 2017.

[5] C. Yu, K. S. Barsim, Q. Kong, and B. Yang, "Multi-level attention model for weakly supervised audio classification," *arXiv preprint arXiv:1803.02353*, 2018.

[6] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[7] S. Hershey, S. Chaudhuri, D. P. Ellis, J. F. Gemmeke, A. Jansen, R. C. Moore, M. Plakal, D. Platt, R. A. Saurous, B. Seybold, *et al.*, "Cnn architectures for large-scale audio classification," in *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*. IEEE, 2017, pp. 131–135.

[8] E. Fonseca, J. Pons, X. Favory, F. Font, D. Bogdanov, A. Ferraro, S. Oramas, A. Porter, and X. Serra, "Freesound datasets: a platform for the creation of open audio datasets," in *Proceedings of the 18th International Society for Music Information Retrieval Conference (ISMIR 2017)*, Suzhou, China, 2017, pp. 486–493.

[9] E. Fonseca, M. Plakal, F. Font, D. P. Ellis, X. Favory, J. Pons, and X. Serra, "General-purpose tagging of freesound audio with audioset labels: Task description, dataset, and baseline," *arXiv preprint arXiv:1807.09902*, 2018.

[10] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," *arXiv preprint arXiv:1710.09412*, 2017.

[11] J. Lee and J. Nam, "Multi-level and multi-scale feature aggregation using pretrained convolutional neural networks for music auto-tagging," *IEEE signal processing letters*, vol. 24, no. 8, pp. 1208–1212, 2017.

[12] X. Meng, B. Leng, and G. Song, "A multi-level weighted representation for person re-identification," in *International Conference on Artificial Neural Networks*. Springer, 2017, pp. 80–88.

ACOUSTIC SCENE CLASSIFICATION USING MULTI-SCALE FEATURES

Liping Yang , Xinxing Chen, Lianjie Tao

Key Laboratory of Optoelectronic Technology and System(Chongqing University), Ministry of Education,
Chongqing University, China
{yanglp, gun_xing, taolianjie007} @cqu.edu.cn

ABSTRACT

Convolutional neural networks(CNNs) has shown tremendous ability in many classification problems, because it could improve classification performance by extracting abstract features. In this paper, we use CNNs to calculate features layer by layer. With the layers deepen, the extracted features become more abstract, but the shallow features are also very useful for classification. So we propose a method that fuses features of different layers(it's called multi-scale features), which can improve performance of acoustic scene classification. In our method, the logMel features of audio signal are used as the input of CNNs. In order to reduce the parameters' number, we use Xception as the foundation network, which is a CNNs with depthwise separable convolution operation (a depthwise convolution followed by a pointwise convolution). And we modify Xception to fuse multi-scale features. We also introduce the focal loss, to further improve classification performance. This method can achieve commendable result, whether the audio recordings are collected by same device(subtask A) or by different devices (subtask B).

Index Terms— Multi-scale features, acoustic scene classification, convolutional neural network, Xception, logMel features

1. INTRODUCTION

Acoustic scene classification is a very complex problem which aim is to recognize the surrounding environment using acoustic signals. It has been used in many filed, such as context-aware services [1], surveillance [2] and robotic navigation [3]. Acoustic scene classification is so important that it has been attracting the attention of researchers in machine learning communities. The consecutive editions of the IEEE AASP Challenges Detection and Classification of Acoustic Scenes and Events(DCASE) [4] release the open and established datasets, and provide the scenario to evaluate and benchmark different approaches for acoustic scene classification and acoustic event detection, which makes the research of acoustic scene classification develop at full speed. Nowadays, many methods have been applied to acoustic scene classification, such as signal processing and machine learning, including dictionary learning [5], matrix factorization [6][7], wavelet filterbanks [8], and recently popular deep learning, such as CNN [9], Gated Recurrent Neural Networks(GRNN) [10].

The general framework for acoustic scene classification usually contains two steps. Fist obtain 2D time-frequency representation of data, and extracting relevant features. Second employ these features to achieve classification. And the most commonly used in acoustic scene classification is the Mel Frequency Cepstral Coefficients (MFCC) [3] and logMel features [18]. Different from a short-time Fourier transform(STFT), the constant Q transformation (CQT) provides a frequency analysis on a log-scale which makes it more adapted to sound and music representations, so the spectrum based on the CQT is also used in acoustic scene classification [19]. After computing the 2D time-frequency representation, some methods have investigated many features that are typically used in computer vision such as histogram of gradients (HOG) [19] and local binary pattern (LBP) [20]. Recently, some researchers have proposed feature learning, and learning features from spectrograms can provide representations that are adapted to the data while addressing the general lack of flexibility of hand-crafted features [6][7].

More recently, methods based on Deep Neural Networks (DNNs) have achieved good performance for acoustic scene classification. In [9], the authors presented a CNNs architecture with localized (small) kernels for environmental sound classification, and proposed data augmentation to overcome the problem of data scarcity. In [21], authors presented a distributed sensor server system for acoustic scene classification in urban environment based on CNNs. To exploit sequential correlation and local spectrum-temporal information, some researchers combined the long short term memory units (LSTM) and CNNs in parallel as lower networks [22].

In this paper, we present a new acoustic scene classification method. We fuse the multi-scale features to improve performance of acoustic scene classification. In order to reduce the number of parameters, we use Xception as the foundation network [11], which is convolutional neural network entirely based on depthwise separable convolution layers, and the Xception architecture is a linear stack of depthwise separable convolution layers with residual connections [11]. We modify the Xception architecture, via taking the output of last three blocks, and global pooling the output of each block. Then concatenate them together to achieve multi-scale feature fusion. The output of each block characterize different features, and the deeper blocks have more abstract features. Considering that features of each block have effect on the acoustic scene classification, we fuse the output of these block, and use multi-scale features to improve classification performance. We also introduce the focal loss [13] to further improve classification performance. Our method can achieve good results on subtask A and subtask B.

The rest of this paper is organized as follows. Section 2 presents modified Xception for acoustic scene classification, and describe how to perform multi-scale feature fusion. Section 3

discusses our experiments and results. Section 4 concludes our work.

2. PROPOSED METHOD

This section introduces the proposed multi-scale features fusion, modified Xception and focal loss of multi-class classification.

2.1. Multi-scale features

CNNs have powerful feature extraction capabilities, which realizes feature extraction and dimensionality reduction through operations such as convolution and pooling. The previous classification methods only used the last feature map, the feature map followed by some fully-connected layers(FC), the FC not only has a large amount of parameters, but also has a large amount of calculation. So at present, the most common methods usually perform global pooling on the last feature map, and then use the softmax layer to achieve classification [12][15].

In image classification, using the last feature map's information only, can achieve great performance. But in our case, its performance is not satisfactory. In the field of object detection, some researchers have used multi-scale feature maps to improve detection performance[16]. Inspired by this idea, we use multi-scale features to improve classification performance. The different feature maps contain different information, these information is helpful for improving the classification performance. In Fig. 1, we illustrated how to fuse the multi-scale features. We use the last three feature maps, and perform global pooling on the features, then concatenate them for fusion.

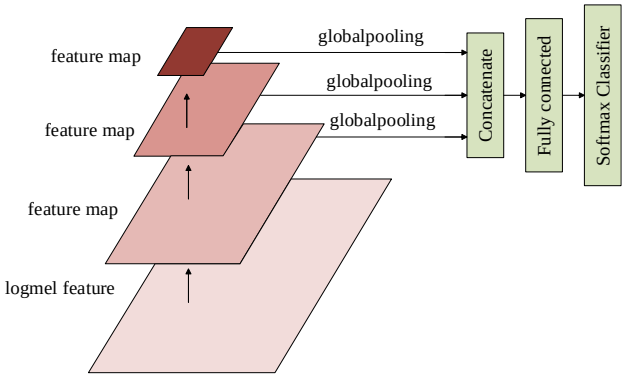


Figure 1: Illustrate how to fuse the multi-scale features

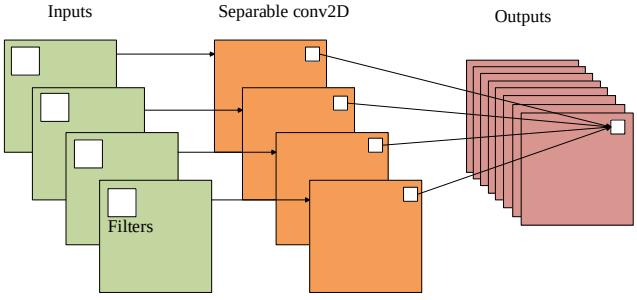


Figure 2: A block of depthwise separable convolution

2.2. Modified Xception

Xception is a convolutional neural network architecture entirely based on depthwise separable convolution layers[12]. In depthwise separable convolution, the convolution operation is split into multiple steps, as shown in Fig. 2. To better illustrate the depthwise separable convolution, we suppose that is a 3×3 size convolutional layer with a 16 channels input and a 32 channels output. The general convolution uses 32 convolution kernels

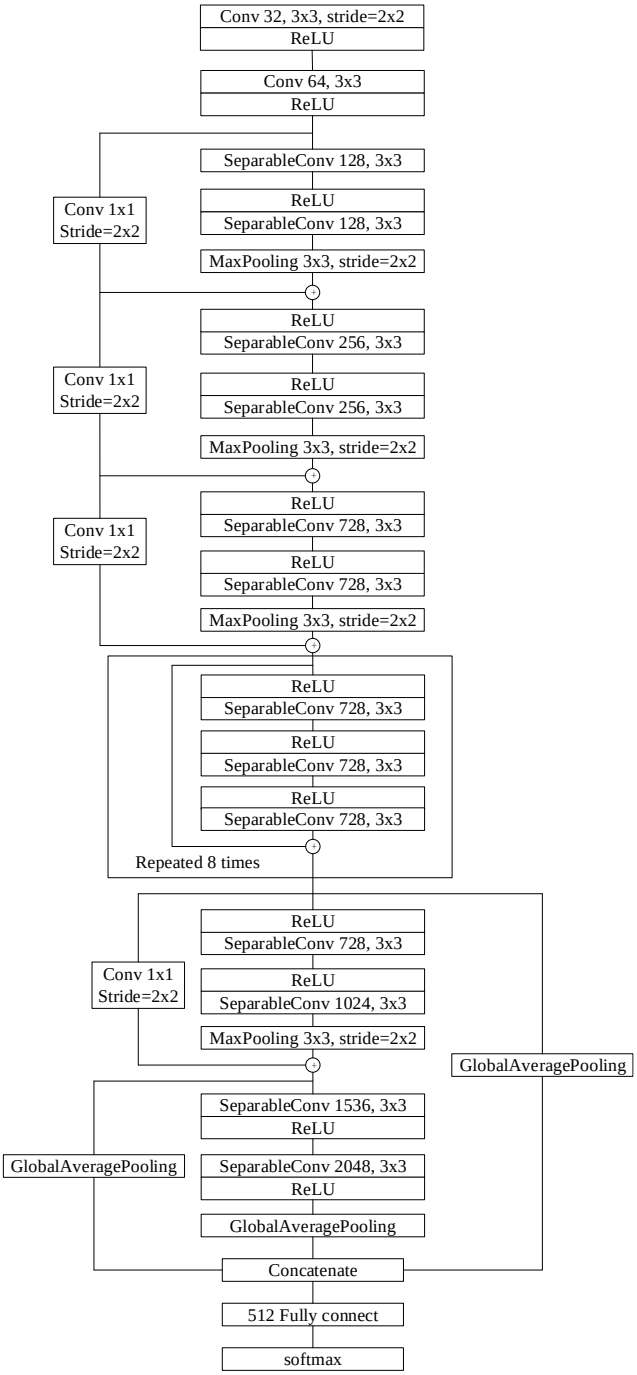


Figure 3: Overview of the modified Xception

convolving with input data, in which $3\times3\times16$ parameters are needed for each convolution kernel. And the output is only one channel. Then 32 convolution kernels need a total of $(3\times3\times16)\times32 = 4068$ parameters.

Depthwise separable convolution is split two steps. First, *depthwise convolution*, which is a spatial convolution performed independently over each channel of one input, 16 convolution kernels (1 channel) of 3×3 size are convoluted with 16 channels input data respectively. Second, *pointwise convolution*, which is a 1x1 (16 channels) convolution, projecting the 32 channels output by the depthwise convolution onto a new channel space. These two steps need $3\times3\times16+(1\times1\times16)\times32 = 656$ parameters, which has less amount of parameters than ordinary convolution. And depthwise separable convolutions are usually implemented without non-linearities activation function.

A complete description of the specifications of the network is given in Fig. 3. The Xception architecture has 36 convolutional layers forming the feature extraction base of the network. The 36 convolutional layers are structured into 14 modules, all of which have linear residual connections around them, except for the first and last modules. We extract the feature maps of the 32nd, 34th, and 36th layers, and perform global pooling on features maps respectively, then concatenate the outputs of global pooling. We fuse the features through FC layer and use softmax layer to perform classification.

2.3. Focal loss

For multi-class classification task, Cross-Entropy (CE) is generally used as the loss function:

$$CE(p,y) = -\sum_{j=0}^{c} y_j \log(p_j) \quad (1)$$

where, p is the model's estimated probability, y is ground-truth class label(one-hot vector), j represents the j -th class. In this paper, we use loss function of acoustic scene classification that is based on CE:

$$l(p,y) = -\frac{1}{n}\sum_{i=0}^{n}\sum_{j=1}^{c} y_j^i \log(p_j^i + \varepsilon) \quad (2)$$

where, y^i represents the label of i -th sample. p^i represents the predicted label of i -th sample, j represents the j -th class. ε is a small positive number to prevent the occurrence of 0 in the logarithmic function.

During the training process, we found, some samples are hard to recognition. These samples would affect the prediction performance of our model. Therefore, we introduce the focal loss [13], the original focal loss start from the CE loss for binary classification. In this paper, we need a multi-class classification loss function, therefore we modify the focal loss. First, we define the probability p_t^i that the i -th sample is predicted correctly:

$$p_t^i = (y^i)^T * p^i \quad (3)$$

where, $(y^i)^T$ represents the transpose of the i -th sample's label, p^i represent the predicted label of i -th sample, $*$ is vector multiplication. The finally loss function are as follows:

$$L(p,y) = -\frac{1}{n}\sum_{i=0}^{n}(1-p_t^i+\varepsilon)^{\gamma}\log(p_t^i+\varepsilon) \quad (4)$$

The modified focal loss, can solve the problem of hard recognition samples, and we only need to select the appropriate hyper parameter γ . Known by definition of the focal loss, for hard to recognize sample, its probability p_t^i is close to 0, and the $(1-p_t^i+\varepsilon)^{\gamma}$ is large. For easy to recognize sample, its probability p_t^i is close to 1, $(1-p_t^i+\varepsilon)^{\gamma}$ is small, so it can down-weight loss of easy sample and up-weight loss of hard sample. It focus training on hard sample.

3. EXPERIMENTAL

3.1. Experimental Setting

We perform experiments on the dataset of DCASE2018 Task1 Subtask A and Subtask B, which consists of 10 scenes, *airport*, *shopping mall*, *metro station*, *street pedestrian*, *public square*, *street traffic*, *tram*, *bus*, *metro* and *park*. We use test set and train set divided by DCASE2018 committee.

Log-scaled mel-spectrogram are used as the input representation of the network. To compute it, the 2-channel wav of subtask A are down mixed to mono, and the wav of subtask B are mono. And STFT is applied using Hamming windows of 4096 samples with 75% overlap. After calculating its power, a mel filter bank is applied consisting of 128 bands. Then we use a filter bank with triangular filters in the frequency domain presenting a peak value of one. Finally, the resulting mel energy values are logarithmically scaled. Resulting log-scaled mel-spectrograms are normalized to zero mean and unit standard deviation for the training set.

The network training was performed by optimizing the focal loss and stochastic gradient descent (SGD) with Nesterov momentum. In the focal loss, γ=3 for subtask A, and γ=1 for subtask B, γ is the optimal values selected by hyper-parameter search. The initial learning rate, and mini-batch size were set to 0.1, and 128, respectively, and use automatic attenuation of learning rate. We train network on dataset for 100 epochs, if the performance of the model is improved after training one epoch, the weight of the model is saved, if the performance of model is not improved after continuous 5 epochs, the learning rate is multiplied by 0.1, and if the performance of model is not improved after continuous 15 epochs, we stop training model.

3.2. Comparison with baselines

Our first experiment compares our method to baseline, the baseline system implements a CNNs based approach, where 40 log mel-band energies are first extracted for each 10-second signal, and a network consisting of two CNNs layers and one fully connected layer is trained to assign scene labels to the audio signals [17]. we perform experiments on development datasets of subtask A and subtask B.

Table 2 presents the results of our proposed method and baseline system. Compared with the baseline system, our proposed method achieves a relative improvement of more than 20%, on subtask A and subtask B.

Table 1: Comparing performances of baseline and our method on the subtask A and subtask B.

Scene	Accuracy(%)			
	Baseline		Our method	
	Subtask A	Subtask B	Subtask A	Subtask B
Airport	72.9	73.3	**77.3**	**78.1**
Bus	62.9	59.4	**84.4**	**88.7**
Metro	51.2	43.3	**79.3**	**72.4**
Metro station	55.4	50.4	**86.8**	**87.8**
Park	79.1	78.1	**86.9**	**91.0**
Public square	40.4	36.2	**51.2**	**53.1**
Shopping mall	49.6	48.2	**88.7**	**79.7**
Street, pedestrian	50.0	51.1	**76.7**	**62.9**
Street, traffic	80.5	80.5	**91.2**	**87.5**
Tram	55.1	51.9	**75.0**	**74.6**
Average	59.7	57.2	**79.8**	**77.6**

Table 2: Analyzing the effects of multi-scale features on the subtask A and subtask B, *w/o* means not using multi-scale features, and *with* means using multi-scale features. In this experiment we don't use the focal loss.

Scene	Accuracy(%)			
	w/o multi-scale features		with multi-scale features	
	Subtask A	Subtask B	Subtask A	Subtask B
Airport	**78.5**	76.4	77.1	**77.8**
Bus	**88.4**	81.7	84.9	**89.0**
Metro	74.3	**73.7**	**78.9**	71.2
Metro station	85.7	85.1	**87.1**	**87.8**
Park	**88.8**	90.7	86.7	**91.2**
Public square	**53.7**	48.0	47.3	**49.8**
Shopping mall	72.7	75.6	**88.6**	**79.2**
Street, pedestrian	65.2	**63.3**	**75.4**	61.4
Street, traffic	86.6	86.2	**92.3**	**87.6**
Tram	74.1	73.1	**74.2**	**74.3**
Average	76.8	75.3	**79.3**	**76.9**

3.3. On the effect of multi-scale features

Our second experiment analyze the effect of multi-scale features on performance. In this experiment, we don't use focal loss, and perform it on development datasets of subtask A and subtask B.

Table 2 presents the results of our proposed method with multi-scale features and without multi-scale features. On subtask A, the method with multi-scale features achieves 2.5% relative improvement compared with the method without multi-scale features, and on subtask B, the improvement is 1.6%. It can been seen that fusion of multi-scale features can improve performance.

3.4. On the effect of focal loss

Our third experiment analyze the effect of focal loss, In this experiment, we use multi-scale features. And perform this experiment on development datasets of subtask A and subtask B.

Table 2 presents the results of our proposed method with focal loss and without focal loss. The focal loss could solve the problem that some samples are difficult to recognize, the method with focal loss achieves 0.6% improvement on subtask A, and 0.7% improvement on subtask B.

Through these experiments, we can draw conclusions, our method can achieve great classification performance on subtask A and subtask B.

Table 3: Analyzing the effects of the focal loss on the subtask A and subtask B, *w/o* means not using focal loss, and *with* means using focal loss. In this experiment, we use multi-scale features.

Scene	Accuracy(%)			
	w/o focal loss		with focal loss	
	Subtask A	Subtask B	Subtask A	Subtask B
Airport	77.1	77.8	**77.3**	**78.1**
Bus	**84.9**	**89.0**	84.4	88.7
Metro	78.9	71.2	**79.3**	**72.4**
Metro station	**87.1**	**87.8**	86.8	**87.8**
Park	86.7	**91.2**	**86.9**	91.0
Public square	47.3	49.8	**51.2**	**53.1**
Shopping mall	88.6	79.2	**88.7**	**79.7**
Street, pedestrian	75.4	61.4	**76.7**	**62.9**
Street, traffic	**92.3**	**87.6**	91.2	87.5
Tram	74.2	74.3	**75.0**	**74.6**
Average	79.3	76.9	**79.8**	**77.6**

4. CONCLUSION

In this paper, we propose an acoustic scene classification method which uses multi-scale features fusion. We use Xception as the foundation network, in order to fuse features, we modify the Xception. This method can achieve great classification performance on subtask A and subtask B. In order to further improve performance, we introduce focal loss of multi-class classification. Although our method is still satisfactory, its biggest problem is the existence of overfitting, and if there are more data to train our model, we would get better performance.

5. REFERENCES

[1] A. J. Eronen et al., "Audio-based context recognition," *IEEE Trans. Audio,Speech, Lang. Process.*, vol. 14, no. 1, pp. 321–329, Jan. 2006.

[2] R. Radhakrishnan, A. Divakaran, and P. Smaragdis, "Audio analysis for surveillance applications," in *Proc. IEEE Workshop Appl. Signal Process.Audio Acoust.*, 2005, pp. 158-161.

[3] S. Chu, S. Narayanan, C.-C. J. Kuo, and M. J. Mataric, "Where am I? Scene recognition for mobile robots using audio features," in *Proc. IEEE Int. Conf. Multimedia Expo*, 2006, pp. 885–888.

[4] A. Mesaros, T. Heittola, A. Diment, B. Elizalde, A. Shah,E. Vincent, B. Raj, and T. Virtanen, "DCASE 2017 challenge setup: Tasks, datasets and baseline system," in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2017 Workshop (DCASE2017)*, November 2017, submitted.

[5] J. Salamon and J. P. Bello, "Unsupervised feature learning for urban sound classification," in *IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, Brisbane, Australia, Apr. 2015, pp. 171–175.

[6] V. Bisot, R. Serizel, S. Essid, and G. Richard, "Acoustic scene classification with matrix factorization for unsupervised feature learning," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICA SSP)*, Shanghai, China, Mar. 2016, pp. 6445–6449.

[7] V. Bisot, R. Serizel, S. Essid, and G. Richard, "Feature learning with matrix factorization applied to acoustic scene classification," in *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 25, no. 6, pp. 1216-1228. June 2017

[8] J. T. Geiger and K. Helwani, "Improving event detection for audio surveillance using gabor filterbank features," in *23rd European Signal Processing Conference (EUSIPCO)*, Nice, France, Aug. 2015, pp. 714–718.

[9] J. Salamon, J. P. Bello, "Deep convolutional neural networks and data augmentation for environmental sound classification," *IEEE Signal Processing Letters*, 2017, pp (99):1-1.

[10] Z. Ren, V. Pandit, K. Qian, et al, "Deep sequential image features for acoustic scene classification," in *Proc. of the Detection and Classification of Acoustic Scenes and Events 2017 Workshop (DCASE2017)*, November 2017, pp. 113-117

[11] F. Chollet, "Xception: deep learning with depthwise separable convolutions." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 1800-1807.

[12] K. He, X. Zhang, S. Ren, and J. Sun. "Deep residual learning for image recognition," *arXiv preprint arXiv:1512.03385*, 2015.

[13] T. Y. Lin, P. Goyal, R. Girshick, et al, "Focal loss for dense object detection," In *Proceedings of the IEEE Conf-erence on Computer Vision and Pattern Recognition*, 2017, pp. 2999-3007.

[14] G. Huang, Z. Liu, et al, "Densely connected convolutional networks," In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 2261-2269.

[15] K. Simonyan, A. Zisserman. "Very deep convolutional networks for large-scale image recognition," In *ICLR*, 2015.

[16] W. Liu, D. Anguelov, et al, "SSD: single shot multibox detector," in *European Conference on Computer Vision*. Springer, Cham, 2016, pp. 21-37.

[17] http://dcase.community/challenge2018/task-acoustic-scene-classification .

[18] S. Park, S. Mun, Y. Lee, et al, "Acoustic scene classification based on convolutional neural network using double image features," in *Proc. of the Detection and Classification of Acoustic Scenes and Events 2017 Workshop (DCASE2017)*, November 2017, pp. 98-102.

[19] A. Rakotomamonjy, G. Gasso, "Histogram of gradients of time-frequency representations for audio scene classification," *IEEE Trans. Audio,Speech, Lang. Process.*, vol. 23, no.1, pp. 142-153, Jan. 2015.

[20] W. Yang, S. Krishnan, "Combining temporal features by local binary pattern for acoustic scene classification," *IEEE Trans. Audio,Speech, Lang. Process.*, vol. 25, no. 6, pp.1315-1324, June 2017.

[21] J. Abeßer, S. I. Mimilakis, et al. "Acoustic scene classification by combining autoencoder-based dimensionality reduction and convolutional neural networks," in *Proc. of the Detection and Classification of Acoustic Scenes and Events 2017 Workshop (DCASE2017)*, November 2017, pp.7-11.

[22] S. H. Bae, I. Choi, and N. S. Kim, "Acoustic scene classification using parallel combination of LSTM and CNN," *Proc. of the Detection and Classification of Acoustic Scenes and Events 2017 Workshop (DCASE2017)*, Sept-ember 2016, pp. 11-15

ACOUSTIC SCENE CLASSIFICATION USING A CONVOLUTIONAL NEURAL NETWORK ENSEMBLE AND NEAREST NEIGHBOR FILTERS

Truc Nguyen[*], *Franz Pernkopf*[†]

Graz University of Technology,
Signal Processing and Speech Communication Lab.,
Inffeldgasse 16c, A-8010 Graz, Austria/Europe,
{t.k.nguyen, pernkopf}@tugraz.at

ABSTRACT

This paper proposes Convolutional Neural Network (CNN) ensembles for acoustic scene classification of tasks 1A and 1B of the DCASE 2018 challenge. We introduce a nearest neighbor filter applied on spectrograms, which allows to emphasize and smooth similar patterns of sound events in a scene. We also propose a variety of CNN models for single-input (SI) and multi-input (MI) channels and three different methods for building a network ensemble. The experimental results show that for task 1A the combination of the MI-CNN structures using both of log-mel features and their nearest neighbor filtering is slightly more effective than the single-input channel CNN models using log-mel features only. This statement is opposite for task 1B. In addition, the ensemble methods improve the accuracy of the system significantly, the best ensemble method is ensemble selection, which achieves 69.3% for task 1A and 63.6% for task 1B. This improves the baseline system by 8.9% and 14.4% for task 1A and 1B, respectively.

Index Terms— DCASE 2018, acoustic scene classification, convolution neural network, nearest neighbor filter.

1. INTRODUCTION

Acoustic scene classification (ASC) is defined as recognition of the environment based on the acoustic scene which is assumed to be a valid characterization of a location or situation. Furthermore, it is assumed to be distinguishable from other scenes based on its acoustic properties [1]. Sound events are introduced as important descriptors for an acoustic scene [2], however,the sound events are complex and can have a high degree of overlap. In real environments, sounds are unstructured and often unpredictable in its occurrence [3] causing more challenges for ASC compared to speech and music signal processing. However, the motivation for recent research on ASC is in designing a system that is able to capture and exploit the specific properties of a given audio scene. These algorithms are embedded in commercial smart devices with microphones to recognize acoustic contextual information.

Up to now, the basic framework of ASC includes feature extraction and classification that have been the crucial stages contributing to the effectiveness of an ASC algorithm. The most popular features applied in the ASC are representations of mel-frequency scales such as mel-frequency cepstral coefficients (MFCCs) and log-mel energies [4], [5]. According to [6], the main reason for their success is that they provide a reasonably good representation of the spectral properties of the signal. Furthermore, a reasonably high inter-class variability allows for class discrimination. Beside that, these features can be used as basis for higher level features. For example, Recurrent Quantification Analysis (RQA) and I-vectors are features obtained from MFCCs by applying recurrent quantification analysis [7] and joint factor analysis (JFA) [4]; Histogram of Gradient (HOG), Linear Binary Pattern (LBP) are well-known image processing techniques that were also used for feature extraction based on various types of spectrograms and MFCCs [8], [9], [10]. Moreover, in order to better cover the characteristics of environmental sounds, low level features such as zero-crossing, spectral centroid, bandwidth, energy have been combined with high level features such as Label Tree Embeding (LTE) [11], [12].

For classification, conventional classifiers such as Gaussian Markov Models (GMMs), Hidden Markov Models (HMMs), Support Vector Machines (SVMs) and Neural Networks (NNs) were applied in almost all submitted reports in DCASE 2013, where no algorithms involving Deep Neural Networks (DNNs) had been used [6]. In DCASE 2016, beside conventional classification methods, many participants applied DNNs such as Convolution Neural Networks (CNNs), Recurrent Neural Networks (RNNs) or combinations of DNNs and GMMs, and HMMs [13], [14] or combinations of CNNs and RNNs [15]. In DCASE 2017 and recent works, deep learning has been even more effective [16], [17], e.g. Generative Adversarial Networks (GANs) have been the most successful system for ASC in DCASE 2017. They have been combined with SVMs for classification [5].

This paper introduces an ASC system which is applied for task 1A and task 1B of the DCASE 2018 challenge. In order to extract more information of the acoustic scene, we use 128 log-mel energies of the spectrogram and additionally apply nearest neighbor filtering (NNF)[18]. Both types of features are considered in the CNNs. All features are preprocessed by splitting the acoustic scene into chunks of 1s. Finally, ensemble methods are applied to combine several features and CNN settings to provide a vote for the 10s data chunks.

The remainder of this paper is organized as follows. Section 2 explains details of the proposed system. Section 3 discuss the experiments and results. Finally, conclusion is provided in Section 4.

2. PROPOSED SYSTEM

The proposed system is illustrated in Fig.1. The system is composed of 3 stages. First, the audio signal is converted to various time-frequency representations in 1s chunks. These features are then fed

[*]Thanks to Vietnamese - Austrian Government Scholarship for funding.
[†]Thanks to Austrian Science Fund.

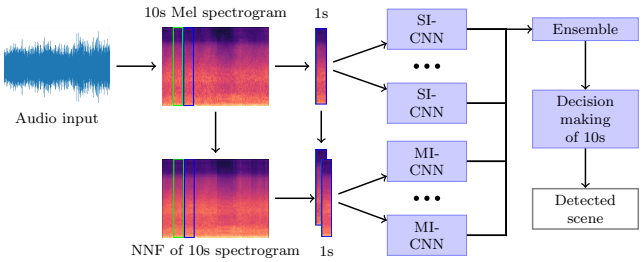


Figure 1: Proposed System

to the CNNs for training the models. Finally, probability outputs of 10 1s chunks of the CNN ensembles are used to produce the scene labels.

2.1. Audio Preprocessing

First, 128 bin mel-energies of the audio input are extracted. According to [16], it is important to keep a sufficient number of bins for representing the spectral characteristics while greatly reducing the feature dimensions. Window size for short-time Fourier transform is selected as 40ms and 20ms for hop size. We keep the sampling rating 48kHz for task 1A and 44.1 kHz for task 1B. In order to generate additional features for MI-CNNs, the mel-spectrogram is processed by a nearest neighbor filter [18]. Both the energies of the spectrogram and the filtered spectrogram are converted into logarithmic scale and are normalized by subtracting the mean value and dividing by the standard deviation. The normalization step is determined feature-wise on the training set and parameters obtained are used to scale both training set and test set. The 10s audio files are processed in 1s audio chunks without overlap and fed to the CNN model as samples.

2.2. Nearest Neighbor Filter

Environmental sounds are often unstructured, neither predictable repetitions nor harmonic sounds [3] that are compounded by sound events and by overlapping of sound events. These sound events could be periodic or randomly repeating sounds such as sounds of a siren, horn of vehicles, sounds of opening and closing metro doors at metro stations etc. Therefore, it is useful for an ASC system to generate features which emphasize the appearance of similar patterns of a sound event in an acoustic scene.

In our ASC system, we use nearest neighbor filters based on Repeating Pattern Extraction Technique (REPET) [18] for cases where repetitions happen intermittently or without a fixed period. The features are processed from spectrograms as follows:

1. Compute a similarity matrix from the frames of spectrogram using a similarity measure such as cosine, euclidean, L1, L2 or manhattan distance.
2. Identify the most similar frames in the spectrogram by using the similarity matrix.
3. Assign the median value of the identified frames for each frequency band to generate the filtered spectrogram.

Empirically, we observed that the euclidean distance is better than cosine distance and the number of nearest-neighbors for each sample is set to 5.[1]

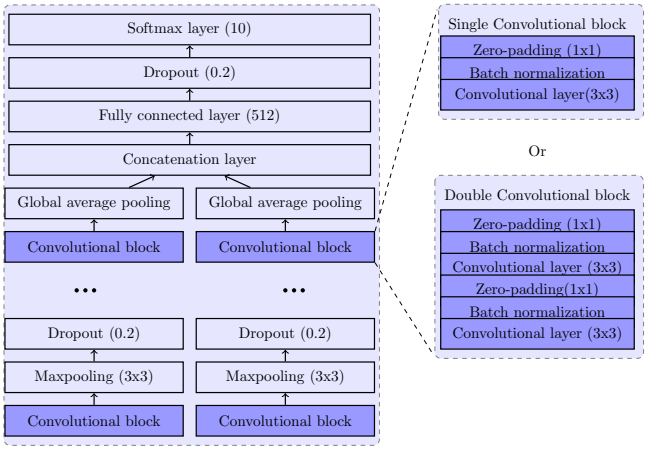


Figure 2: MI-CNN with single and double convolutional blocks

2.3. Multi-input Convolution Neural Network

MI-CNNs have been used for ASC with different input features or structures of each branch of the CNN architecture. For example, in [20], authors used their CNN model as a "parallel" CNN architecture with different filter sizes and max-pooling sizes. In [15], they used a combination of Long Short Term Memories (LSTMs) and CNNs as a feature extraction step for each branch of their model. In addition, according to [16], their CNN model used left-right (LR), *L+R* and *L-R* (MS), or harmonic-percussive source separation pairs as different input sources.

Our MI-CNN is inspired by these works. We feed 128 log-mel energies to one input branch of the CNN and their nearest neighbor filtered version to another one with the same CNN structure. Subsequently, we concatenate both branches before the fully-connected layer. Because the size of each sample is small i.e. 128 bins x 50 frames, 1x1 zero-padding is added to each convolution step in order to ensure that the whole data is processed. We proposed to use either a single convolutional block or a double convolutional blocks. A convolutional block consists of zero-padding, batch normalization and convolution layers, in which Rectifier Linear Units (ReLUs) are used as activation function. The single/ double convolutional block is followed by a max-pooling layer and a dropout layer for the purpose of reducing dimensionality of the convolutional output and to ease the computation for upper layers as well as to reduce over-fitting in the training phase. Specifically, the last convolution blocks of the input branches are followed by global average pooling (GAP) instead of max-pooling and dropout.

CNNs have been considered as an extractor of high-level features and different structures of CNNs learn different high-level features. In this research, we create a diversity of CNN structures by adjusting the depth of the CNNs as well as the structure of convolutional blocks through various number of single convolutional blocks and double convolutional blocks. Beside that, the diversity of CNN structures is enriched by using single-input channel (SI) CNNs, i.e., using only one input branch. The structures of the MI-CNN using single and double convolutional blocks are shown in Fig. 2.

[1]The processing is done by using Librosa toolbox `https://librosa.github.io/librosa`

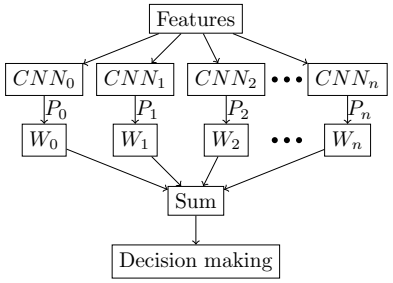


Figure 3: Architecture of CNN ensemble.

Empirically, we select the number of filters of the convolutional layers for the CNNs including 2, 3 and 4 single or double convolutional blocks at 32 - 256, 32 - 64 - 256 and 32 - 64 - 128 - 256, respectively. Both convolutional layers of each double convolutional block have the same number of filters. The number of parameters of the proposed CNN models are shown in Table 1 and they are same for both tasks.[2]

2.4. Convolutional Neural Network Ensemble

Ensembles of CNNs combine the output probabilities of CNNs in order to improve performance [21]. The CNNs in the ensemble are trained individually and then their outputs are combined by majority voting, averaging, weighed averaging or model selection with and without replacement [22].

We compared performance of three ensemble methods named average ensemble (AE), weighted averaging ensemble (WE) and ensemble selection with replacement (ES). Basically, the similarity of these ensemble methods is that the output probabilities from all CNNs are averaged before making predictions. However, they are different in determining the contribution levels of each model to the ensemble using weights. Fig.3 shows the general architecture of the ensemble. Average ensemble is a simple ensemble where the output probabilities from all CNNs are equally weighted and averaged. The constraint of the weights is to be equal for all CNNs and sum to one. Weighted averaging ensemble and ensemble selection are more complex. Weighted averaging ensemble determines the optimal weights by minimization of the cross-entropy loss of ground-truth labels and estimated labels with constraints of the weights to sum to one. Ensemble selection with replacement [22] is an iterative method that allows models to be added to an ensemble multiple times such that the performance of the combination is maximized. The model weights are equivalent to the number of times of the model has been selected divided by the total number of models in the ensemble.

We use the test data to determine the optimal weights for WE and ES. Sequential Least Squares Programming (SLSQP) is used for optimization of WE. For ES, we start with the best model among 12 candidate models in the ensemble before greedy step-wise selection of 200 iterations is performed. The number of selections of each model in the proposed ensemble ES is listed in Table 1 for task 1A and task 1B. There is a significant difference between the weight values of WE and ES. These weights are used for evaluation.

[2]The CNNs are implemented on Keras `https://github.com/keras-team/keras`

Table 1: Number of parameters of the proposed models and number of times the models have been selected by ensemble selection in task 1A and 1B.

Algorithms	Parameters	Task1A	Task 1B
SI_s_2cnn_D	211718	1	14
SI_s_3cnn_D	304010	8	9
SI_s_4cnn_D	525350	1	21
SI_db_2cnn_D	811770	33	51
SI_db_3cnn_D	941074	3	45
SI_db_4cnn_D	1310042	4	13
MI_s_2cnn_D	417794	48	21
MI_s_3cnn_D	602378	0	4
MI_s_4cnn_D	1045058	3	10
MI_db_2cnn_D	1617898	28	3
MI_db_3cnn_D	1876506	6	10
MI_db_4cnn_D	2614442	66	0
Sum	12278040	201	201

In addition, we try majority voting (MV) in which the output probabilities of every 1s chunk is binarized to “0” and “1” with the global threshold at 0.5. Majority voting determines the class which occurs most often among 10 1s chunks of an audio file. For average voting (AV) we use the *argmax* on the mean of the probabilities over 10 s. The experimental results show that AV nearly always outperforms MV.

3. EXPERIMENTS

3.1. Data

The audio dataset for the ASC task of DCASE 2018 includes two different versions, TUT Urban Acoustic Scene 2018 and TUT Urban Acoustic Scene 2018 Mobile recorded in six European cities for 10 scenes. The former dataset is used for task 1A where the development and evaluation data are recorded by the same device. While the later one is used for task 1B in which the development set is comprised of task 1A dataset resampled and averaged into a single channel and a small amount of data is recorded by other devices. The original recordings were split into 10-second segments that are provided in the individual files.

The task 1A dataset includes 8640 segments with 6122 segments for training and 2518 segments for testing. The task 1B training subset contains 6122 segments from device A, 540 segments from device B, and 540 segments from device C. The test subset contains 2518 segments from device A, 180 segments from device B, and 180 segments from device C.

3.2. Setup

The validation set accounts for approximately 30% of the original training data. We use a balancing mode for separation such that there are no segments from the same location and city in both training and validation data sets. Acoustic features are log mel-band energies of 128 frequency bands and their nearest neighbor filtered version with 40 ms analysis frame and 50% hop size. The network training is carried out by optimizing the categorical cross-entropy and the Adam optimizer at learning rate of 0.001 is used. We use Glorot uniform data to initialize the network weights. The number of epochs and batch size was 500 and 16, respectively, and data is

Table 2: Accuracy of the proposed models and of the ensemble methods using majority voting and average voting with and without dropout.

Algorithms	1A_MV	1A_AV	1B_MV	1B_AV
Baseline	59.7 ± 0.7	-	45.6 ± 3.6	-
SI_s_2cnn_NoD	61.1	62.3	54.2	56.1
SI_s_3cnn_NoD	64.3	65.0	56.9	57.5
SI_s_4cnn_NoD	63.9	64.7	53.1	54.4
SI_db_2cnn_NoD	63.6	64.4	57.5	58.9
SI_db_3cnn_NoD	63.0	64.1	59.2	60.6
SI_db_4cnn_NoD	64.3	65.3	51.4	53.6
MI_s_2cnn_NoD	61.0	62.1	51.1	52.2
MI_s_3cnn_NoD	64.5	64.4	54.2	55.3
MI_s_4cnn_NoD	62.7	63.4	54.2	54.7
MI_db_2cnn_NoD	66.3	66.8	53.6	55.6
MI_db_3cnn_NoD	63.6	64.0	57.5	56.4
MI_db_4cnn_NoD	63.1	63.2	52.8	52.5
AE_NoD	62.7	66.8	54.4	62.2
WE_NoD	63.4	66.9	54.2	62.5
ES_NoD	63.8	68.5	52.5	63.1
SI_s_2cnn_D	62.7	63.5	57.8	57.8
SI_s_3cnn_D	65.4	65.6	58.1	58.3
SI_s_4cnn_D	63.1	62.9	54.7	55.8
SI_db_2cnn_D	64.3	64.5	60.3	62.2
SI_db_3cnn_D	64.9	65.2	54.4	55.8
SI_db_4cnn_D	64.3	64.6	53.1	54.4
MI_s_2cnn_D	63.8	64.4	54.2	56.9
MI_s_3cnn_D	63.9	64.4	52.8	53.9
MI_s_4cnn_D	61.9	62.6	56.7	56.4
MI_db_2cnn_D	63.5	64.0	55.0	54.4
MI_db_3cnn_D	64.3	64.3	55.3	56.1
MI_db_4cnn_D	65.2	65.8	52.5	53.1
AE_D	63.5	67.4	53.9	61.4
WE_D	65.3	68.3	54.2	61.7
ES_D	65.5	**69.3**	56.7	**63.6**

shuffled between epochs. Model performance is evaluated on the validation set after each epoch and the selected model is the best performing one on the validation set.[3]

3.3. Performance on the test set

Table 2 presents the accuracy of task 1A and task 1B for the different SI-CNNs (SI_) and MI-CNNs (MI_) using majority voting (_MV) and average voting (_AV). The CNNs consists of various numbers of single convolutional blocks (_s) or double convolutional blocks (_db) as well as dropout layers (_D) and no dropout layers (_NoD). The performances of different ensemble methods of the 12 models are also presented. For determining the weights of ES and WE the labels of the test set are used.

According to the results of Table 2, we can see that systems using the average voting method almost always performs better compared to majority voting. Results of average ensemble (AE_) and weighted average ensemble (WE_) are nearly the same and lower than of ensemble selection (ES_). Furthermore, dropout slightly improves the performances.

CNN models using double convolutional blocks (_db) are not always better than CNNs using single convolutional blocks (_s). Most of the (_db) CNN models get higher accuracy compared to the (_s) CNN models for task 1B while most of performances of the (_s) CNN models are better than that of the (_db) CNN models for task 1A.

Moreover, Table 2 shows that NNF features are not really helpul for individual MI_CNN models since most of individual MI_CNNs get lower accuracy than individual SI_CNN models for both tasks. However, they are useful for our ensemble system. Particularly, we can see from Table 1, MI_CNNs using NNF features contribute about three quater among all model components to build the ensemble (ES_) for task 1A but they accupy approximately one quater of the model components of the ensemble (ES_) for task 1B. Feature characteristics that are extracted from different devices' recording files of task 1B dataset are more complex than that of task 1A. So complicated models i.e., the (MI_db) CNNs tend to overfit for task 1B.

The differrent submissions for task 1A and task 1B are 1A_ES_D and 1B_ES_D that use average voting of ensemble selections with dropout . Class-wise accuracy of both are represented in table 2.

Table 3: Class-wise accuracy of submissions on the test set for task 1A and 1B.

Algorithms	1A_ES_D	1B_ES_D
Airport	75.8	58.3
Bus	73.1	80.6
Metro	57.9	41.7
Metro station	76.1	61.1
Park	83.9	91.7
Public square	58.3	55.6
Shopping mall	41.9	75.0
Street_pedestrian	57.5	50.0
Street_traffic	88.6	83.3
Tram	80.1	38.9
Average	**69.3**	**63.6**

4. CONCLUSION

In this paper, we proposed ensembles of 12 CNN structures in order to enhance the classification accuracy for task 1A and task 1B of DCASE 2018 challenge. We also introduce nearest neighbor filtering for MI-CNN structures, which emphasizes the sound events in a scene. Although the new features are not really strong for individual MI-CNNs, our proposed ensemble system significantly improves over the baseline system for all datasets and achieved 69.3% and 69.0% for task 1A and 1B on the evaluation set, respectively. The proposed system was ranked first for task 1B of the DCASE 2018 challenge.

5. ACKNOWLEDGMENT

This research was supported by Vietnamese - Austrian Government scholarship and by the Austrian Science Fund (FWF) under the project number I2706-N31. We acknowledge NVIDIA for providing GPU computing resources.

[3]Thanks to the DCASE organizers for providing the baseline system source code and the DCASE-UTIL toolbox `https://github.com/DCASE-REPO`

6. REFERENCES

[1] A. Mesaros, T. Heittola, A. Diment, B. Elizalder, A. Shah, E. Vincent, B. Raj, T. Virtanen, " DCASE 2017 challenge setup: Tasks, datasets and baseline system," DCASE2017 Challenge, Tech. Rep., Nov. 2017.

[2] T. Heittola, A. Mesaros, A. Eronen, T. Virtanen, "Context-dependent sound event detection," *EURASIP Journal on Audio, Speech and Music Proceeding*, 2013.

[3] S. Chu, S. Narayanan and C. Jay Kuo "Content analysis for acoustic environment classification in Mobile Robots," *Proc. AAAI Fall Symposium*, Oct. 2006.

[4] H. Eghbal-zadeh, B. Lehner, M. Dorfer and G. Widmer , "CP-JKU Submissions for DCASE-2016: a Hybrid Approach Using Binaural I-Vectors and Deep Convolutional Neural Networks," DCASE2016 Challenge, Tech. Rep. Sep. 2016.

[5] S.Mun, S. Park, D. Han and H. Ko, "Generative Adversarial Network Based Acoustic Scene Training Set Augmentation and Selection Using SVM Hyper-Plane," DCASE2016 Challenge, Tech. Rep. Sep. 2017.

[6] A. Mesaros, T. Heittola, E. Benetos, P. Foster, M. Lagrange, T. Virtanen and M. D. Plumbley, " Detection and Classification of Acoustic Scenes and Events: Outcome of the DCASE 2016 Challenge," *IEEE/ACM Trans. Audio, Speech, and Language Process.*, vol. 26, pp. 379-393, Nov. 2017.

[7] G. Roma, W. Nogueira, P. Herrera, and R. De-boronat, "Recurrence quantification analysis features for auditory scene classification," *Proc. IEEE AASP*, 2013, pp. 1 - 4.

[8] V. Bisot, S. Essid and G. Richard, "HOG and subband power distribution image features for acoustic scene classification," *Proc. EUSIPCO*, pp. 719-723, 2015.

[9] A. Rakotomamonjy and G. Gasso, "Histogram of Gradients of Time-Frequency Representations for Audio Scene Classification," *IEEE/ACM Trans. Audio, Speech, and Language Process.* , vol. 23, no. 1, pp. 142-153, 2015.

[10] S. Abidin, R. Togneri and F. Sohel, "Enhanced LBP texture features from time frequency representations for acoustic scene classification," *Proc.IEEE ICASSP* 2017, pp. 626-630.

[11] H. Phan, L. Hertel, M. Maass, P. Koch and A. Mertins, "CNN-LTE: a Class of 1-X Pooling Convolutional Neural Networks on Label Tree Embeddings for Audio Scene Recognition," DCASE2016 Challenge, Tech. Rep. Sep. 2016.

[12] A. Dang, T. H. Vu and J. .C. Wang, "Acoustic scene classification using convolutional neural networks and multi-scale multi-feature extraction," *Proc. ICCE*, pp. 1-4, 2018.

[13] G. Takahashi, T. Yamada, S. Makino and N. Ono, "Acoustic Scene Classification Using Deep Neural Network and Frame-Concatenated Acoustic Feature," DCASE2016 Challenge, Tech. Rep., Sep. 2016.

[14] G. Marques and T. Langlois, "TUT Acoustic Scene Classification Submission," DCASE2016 Challenge, Tech. Rep., Sep. 2016.

[15] S. Bae, I. Choi and N. Kim, "Acoustic Scene Classification Using Parallel Combination of LSTM and CNN," DCASE2016 Challenge, Tech. Rep. Sep. 2016.

[16] Y.Han and J. Park, " Convolutional Neural Networks with Binaural Representations and Background Subtraction for Acoustic Scene Classification," DCASE2017 Challenge, Tech. Rep. Sep. 2017.

[17] B. Lehner,H. Eghbal-zadeh, M. Dorfer, F Koreniowski, K. Koutini and G. Widmer, " Classifying Short Acoustic Scenes with I-Vectors and CNNs: Challenges and Optimisations for the 2017 DCASE ASC Task," DCASE2017 Challenge, Tech. Rep. Sep. 2017.

[18] Z. Rafii and B. Pardo,"Music/voice separation using the similarity matrix," *Proc. ISMIR* , pp. 583-588, 2012.

[19] M. Brian, C. Raffel, D. Liang, D. Ellis, M. McVicar, E. Battenberg, and O. Nieto, "Librosa: Audio and music signal analysis in python," in*Proc. the 14th Python in Science*, pp. 18-25. 2015.

[20] T. Lidy and A. Shindler, "CQT-Based Convolutional Neural Networks for Audio Scene Classification and Domestic Audio Tagging," DCASE2016 Challenge, Tech. Rep. Sep. 2016.

[21] X.Frazao and L. Alexandre, " Weighted Convolutional neural Network Ensemble," Bayro-Corrochano E., Hancock E. (eds) Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications. CIARP 2014. Lecture Notes in Computer Science, vol 8827. Springer, Cham.

[22] R. Caruana, A. Niculescu-Mizel, G. Crew and A. Ksikes, " Ensemble Selection from Libraries of Models," *Proc. ICML*, pp. 18-, Canada, 2004.

ATTENTION-BASED CONVOLUTIONAL NEURAL NETWORKS FOR ACOUSTIC SCENE CLASSIFICATION

Zhao Ren[1], *Qiuqiang Kong*[2], *Kun Qian*[1], *Mark D. Plumbley*[2], *Björn W. Schuller*[1,3]

[1] ZD.B Chair of Embedded Intelligence for Health Care and Wellbeing, University of Augsburg, Germany
[2] Centre for Vision, Speech and Signal Processing (CVSSP), University of Surrey, UK
[3] GLAM – Group on Language, Audio & Music, Imperial College London, UK

zhao.ren@informatik.uni-augsburg.de, schuller@ieee.org

ABSTRACT

We propose a convolutional neural network (CNN) model based on an attention pooling method to classify ten different acoustic scenes, participating in the acoustic scene classification task of the IEEE AASP Challenge on Detection and Classification of Acoustic Scenes and Events (DCASE 2018), which includes data from one device (subtask A) and data from three different devices (subtask B). The log mel spectrogram images of the audio waves are first forwarded to convolutional layers, and then fed into an attention pooling layer to reduce the feature dimension and achieve classification. From attention perspective, we build a weighted evaluation of the features, instead of simple max pooling or average pooling. On the official development set of the challenge, the best accuracy of subtask A is 72.6 %, which is an improvement of 12.9 % when compared with the official baseline ($p < .001$ in a one-tailed z-test). For subtask B, the best result of our attention-based CNN is a significant improvement of the baseline as well, in which the accuracies are 71.8 %, 58.3 %, and 58.3 % for the three devices A to C ($p < .001$ for device A, $p < .01$ for device B, and $p < .05$ for device C).

Index Terms— Acoustic Scene Classification, Convolutional Neural Network, Attention Pooling, Log Mel Spectrogram

1. INTRODUCTION

Acoustic scene classification, as a subfield of computational auditory scene analysis (CASA) [1], aims at enabling devices to recognise the acoustic environment. It has been successfully employed in a series of applications, including intelligent wearable interfaces [2, 3], smartphone navigation systems [4], context-aware computation [5], and many more. In the field of machine learning, a number of models have been applied in the past 'detection and classification of acoustic scenes and events' (DCASE) challenges, such as support vector machines [6, 7], hidden markov models [8], autoencoders [9], and convolutional neural networks (CNN) [10, 11]. In the acoustic scene classification task of the IEEE AASP Challenge in this year [12], researchers are provided with the opportunity to investigate training a robust model on a dataset from multiple recording devices. The acoustic scene classification task in this challenge includes two subtasks with different data sources – one is based on single recording device, and the other is based on three devices. The dataset has been divided into a 'development set' with a training/test partitioning, and a non-public evaluation set. Both of the subtasks require participants to classify the acoustic data into ten classes of scenes.

In recent years, time-frequency transformation images have shown their superiority in improving the performance in the acoustic scene classification task [13]. Different kinds of time-frequency transformation images have been applied for feature extraction, such as Constant-Q-Transform (CQT) spectrogram [13], Short-Time Fourier Transformation (STFT)-based spectrogram [9], scalogram [10], and log mel spectrogram [14]. In this paper, we use a log mel spectrogram image representation as it performed excellent in the acoustic scene classification task of DCASE 2017 [15].

With log mel spectrogram images, we construct an end-to-end CNN model for classification. A number of CNNs have been presented successfully in image processing, particularly in the ImageNet Large Scale Visual Recognition Challenge [16]. Compared with the dataset with around several hundreds of thousands samples in that challenge, the dataset in DCASE challenge contains less than ten thousands of audio waves for training. In this regard, CNNs with relatively more shallow layers than the CNNs for ImageNet, are utilised in our work, including AlexNet and VGG with four convolutional layers. In addition, a CNN topology with different structure and kernel size is designed to improve the performance.

To avoid over fitting caused by the large size of the feature maps that are obtained after the convolutional layers in the CNN, pooling usually serves to compute features by reducing the feature dimension. Max pooling and average pooling are the most frequently employed pooling models to obtain smaller feature dimension. Max pooling is achieved through extracting the largest value inside a filter as parameters of the max pooling layer [17, 18], and average pooling aims at obtaining the average value of a filter as its parameter [19]. Unfortunately, both of these pooling models cannot utilise each feature reasonably according to its contribution. Max pooling ignores other potentially helpful features besides the feature with the maximum value; average pooling treats each feature equally, easily leading to some suboptimal results because of the interference from useless features. To solve this problem, an attention model, attempting to compute the contribution of each feature, was proposed and utilised in many applications, including natural language processing [20], visual question answering [21], and even audio classification [22].

The two main contributions of our paper for acoustic scene classification are as follows. The first is that we design an end-to-end CNN to train the model, and compare it with the state-of-the-art CNN models. Second, we propose an attention-based CNN by weighting the contribution of each feature and explaining this model from a probability perspective in multiple instance learning [23].

The remainder of this paper is structured as follows: in Section 2, we describe the proposed approach, the pipeline of which is shown in Figure 1, the database description, experimental set up, and results, are presented in Section 3; finally, conclusions are given in Section 4.

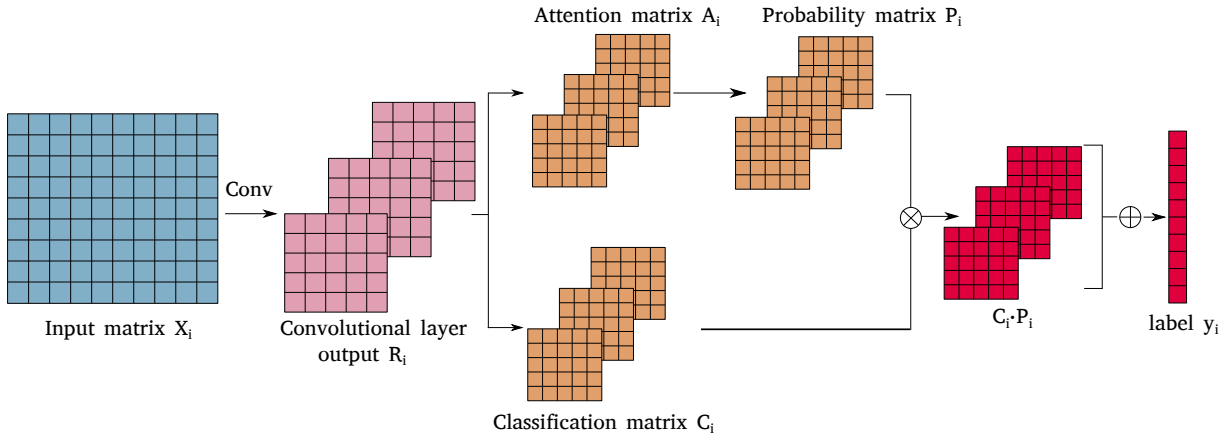


Figure 1: The framework of our proposed attention pooling system. First, the output of the convolutional layers R_i is obtained through the convolutional layers when X_i is the input matrix. The attention matrix A_i and the classification matrix C_i with the class number of the channels are then generated from R_i. Further, the probability P_i calculated from A_i multiplies with C_i for an element-wise product $C_i \cdot P_i$. The prediction y_i is finally obtained by summing up of $C_i \cdot P_i$.

Table 1: Configurations of the convolutional neural networks. Convolutional layers are denoted as 'the number of convolution layers × conv(receptive field size – number of channels)' with the stride 's(stride size)'.

AlexNet	VGG-4	our CNN
Input: image X_i $(1 \times m \times n)$		
1×conv11-64; s1 Maxpooling	1×conv3-64; s1 Maxpooling	1×conv5-64; s2
1×conv5-192; s1 Maxpooling	1×conv3-128; s1 Maxpooling	1×conv5-128; s2
1×conv3-384; s1 2×conv3-256; s1 Maxpooling	1×conv3-256; s1 1×conv3-512; s1 Maxpooling	1×conv5-256; s2 1×conv5-512; s2
Output: R_i $(c \times m' \times n')$		

2. METHODOLOGY

In this section, we describe our proposed neural network in two subsections. The first is dedicated to the convolutional neural network, and then the pooling models following classification will be introduced.

2.1. Convolutional Neural Networks

Given the successful application of image-based neural networks for acoustic scene classification tasks in [10, 24], we employ three end-to-end CNN topologies in this work, including 'AlexNet' [25], 'VGG-4' [26], and 'our CNN' with a different structure from the former two CNNs, as shown in Table 1.

The log mel spectrogram images with one channel are first extracted from audio waves and are fed into the CNN model. The $i-th$ image $X_i, i = 1, ..., N$ with size of $1 \times m \times n$, in which N is the number of images, could generate an output $R_i, i = 1, ..., N$ with size of $c \times m' \times n'$ by the convolutional layers of the CNN, where c means the number of channels. In Table 1, the AlexNet model uses the same parameters as the original AlexNet [25]. The conventional VGG structures [26] are not used in our work, since the DCASE dataset is much smaller than the ImageNet database [27]. Thereby, we design a VGG-4 with four convolutional layers and using the same kernel size of three with the typical VGG model for each layer. Additionally, a CNN without max pooling during convolution, but with a stride with size of 2, is designed to weaken the effect of max pooling, and with a kernel size inbetween AlexNet and VGG-4 to explore the effect of the kernel size in order to reach a better performance.

Please note that the rectified linear unit activation function 'ReLU' is applied for each convolutional layer. Different from the typical AlexNet or VGG, batch normalisation is employed for all convolutional layers in our work, as this can accelerate deep networks and improve the performance of CNNs [28, 29].

The output of the convolutional layers R_i has the size of $c \times m' \times n'$, which is not appropriate for classification of ten classes as demanded in our case due to its large size (2 000 for AlexNet, and at least 400 000 for VGG-4 and our CNN). Therefore, a pooling mechanism is employed to reduce the feature dimension in the next subsection.

2.2. Pooling Mechanism

The output R_i of each spectrogram image by the CNN, which could be viewed as a bag of instances, contains c feature maps with m' feature vectors at n' time steps. The matrix with size of (m', n') is an instance in a bag. Based on multi-instance learning [30, 23], we consider that the classification model is given a number of pairs $\{(R_i, y_i)\}, i = 1, ..., N$. For the matrix R_i, its correspondent label is $y_i \in \{0, 1\}^L$, where L is the number of the scene classes. To achieve the classification, it is necessary to reduce the dimension of R_i from three to single channel dimension. In this subsection, two traditional pooling methods, max pooling and average pooling, and our proposed attention pooling will be introduced.

2.2.1. Max Pooling

On the assumption that the maximum classification value of each instance is the prediction of a bag [31], the max pooling model is described as

$$R_i^* = \max_{1<q<n'} \max_{1<p<m'} R_i, \tag{1}$$

where R_i^* is a feature vector prepared to classify the labels by linear transformation. Max pooling has been widely applied in CNNs for image classification and performs well [27], but sometimes unsatisfactory as it loses the time and location information when only choosing the maximum value at the dimensions of the time steps and feature vectors.

2.2.2. Average Pooling

Based on the collective assumption in [32], we assume that all instances contribute equally for the prediction in a bag. Accordingly, the definition of average pooling is

$$R_i^* = \frac{1}{m'n'} \sum_{1<q<n'} \sum_{1<p<m'} R_i. \tag{2}$$

As average pooling weights the contribution of each instance equally, unfortunately, it is possible that it diminishes the effect of some important features and augments some noisy features, leading to potentially imperfect prediction results.

2.2.3. Attention Pooling

As mentioned, both the max and average pooling models cannot calculate fairly according to the contribution of each instance. Therefore, computing the contributions of instances in a bag, which aims to obtain the weight of each instance, is a challenging task. To solve this problem, an attention-based pooling model is proposed for the aimed at acoustic scene classification task, as shown in Figure 1.

As to the matrix R_i, we feed R_i twice into two parallel 1-by-1 convolution layers with the channel number of L and a kernel size of 1, to reduce the dimension of the feature maps to prepare for classification. Both convolutional layers are followed by activations, including sigmoid activation for the attention matrix A_i, and 'log softmax' for the classification matrix C_i. Therefore, two matrices A_i and C_i with size (L, m', n') are obtained. To compute the contribution of each instance, in other words the contributions of elements in each feature map, the probability matrix P_i is defined as

$$P_i = A_i / \sum_{1<q<n'} \sum_{1<p<m'} A_i. \tag{3}$$

With the probability matrix P_i, which holds the weight of each element of C_i, the prediction y_i is

$$y_i = \sum_{1<q<n'} \sum_{1<p<m'} C_i \cdot P_i. \tag{4}$$

Finally, the predicted label is obtained as the summing up of the element wise product of C_i and P_i.

Attention pooling is capable to overcome the disadvantage of max pooling and average pooling, weighting the contribution for each instance through a 1-by-1 convolutional layer. As shown in Figure 2, attention pooling can give weights for instances according to their contributions, thereby achieving more optimal prediction results.

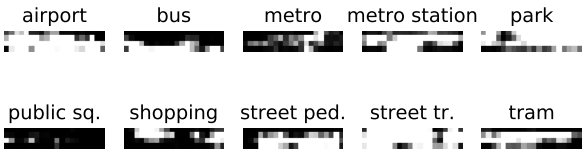


Figure 2: Heat maps of each scene class computed by the matrix P_i in our attention-based CNN. The heat map in this figure is the transpose matrix of P_i with a size of $(20, 4)$ for a better display. The horizontal axis represents the time steps, and the vertical axis the feature vectors.

3. EXPERIMENTAL RESULTS

3.1. Database

Our proposed approach is evaluated on the development dataset of the acoustic scene classification task in DCASE 2018 [12]. The dataset was recorded in various scene environments, and several locations for each scene. Each original recording with a length of 5-6 minutes was segmented into clips of 10 seconds. The sampling rate is set to 44.1 kHz in our work. The dataset contains 10 scene classes, including *airport*, *shopping mall*, *metro station*, *street pedestrian*, *public square*, *street traffic*, *tram*, *bus*, *metro*, and *park*. The task consists of two subtasks according to different recording devices, which comprise Soundman OKM II Klassik/studio A3, an electret binaural microphone, and a Zoom F8 (referred to device A), and two customer devices, e. g. smartphones, cameras, (referred to device B and C), thereby two sub-datasets are provided:

1) TUT Urban Acoustic Scenes 2018, was recorded by device A with 8 640 segments in total.

2) TUT Urban Acoustic Scenes 2018 Mobile, contains recordings from devices A, B, and C. In this dataset, the recordings are made up of 8 640 audio files from device A, and 720 audio files by device B and C in parallel.

3.2. Experimental Setup

The log mel spectrogram images are firstly extracted from each audio wave, with a Hamming window size of 2 048, overlap of 672, and 64 mel bands. Therefore, a feature map with a size of (320, 64) is generated for each audio file. The features are then fed into CNNs as mentioned in Section 2, using the 'Adam' optimiser with a learning rate of 0.001. The CNNs are optimised during 3000 maximum iteration steps, which are empirically set. As the accuracy on test set floats in a small interval after convergence, the iteration step corresponding to the highest accuracy during all iterations is chosen as the step where the training is stopped. The CNN architectures are implemented using Pytorch[1].

3.3. Results and Discussion

The results evaluated on the development set are shown in Table 2. We can see that, nearly all of our pooling models achieve improvements compared to the official baseline system. The attention pooling model performs better than max and average pooling models at AlexNet and our CNN. However, the attention pooling model at

[1] https://pytorch.org/

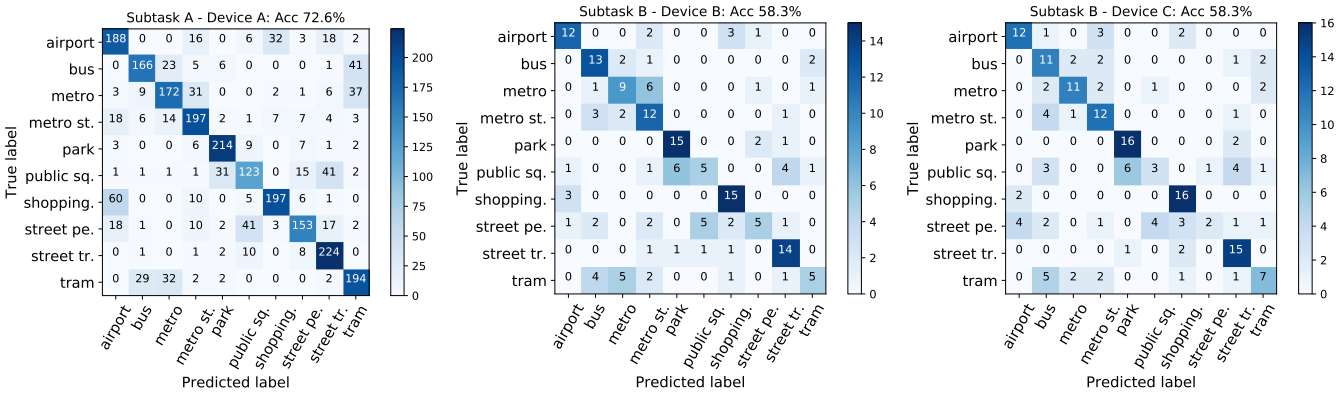


Figure 3: Confusion matrices of device A in subtask A and device B and C in subtask B by the best model. Our proposed CNN with attention pooling is the best model for both subtask A (SUBA) and subtask B (SUBB).

Table 2: Performance comparison of the baseline and CNN topologies of AlexNet, VGG-4, and our CNN, with three pooling models – max, average, and attention, evaluated on the official development set of Subtask A (SUBA) and Subtask B (SUBB). The dataset recorded by device A is employed for the evaluation of Subtask A, and the datasets from device A, B, and C are used for Subtask B. $\overline{(B,C)}$ stands for the mean evaluation result of device B and C. The experimental results are evaluated by accuracy [%].

		SUBA	**SUBB**			
NN	Pooling	A	A	B	C	$\overline{(B,C)}$
Baseline		59.7	58.9	45.1	46.2	45.6
AlexNet	max	67.2	62.2	51.7	54.4	53.1
AlexNet	average	64.3	60.8	51.1	52.2	51.7
AlexNet	attention	67.2	64.2	53.3	46.7	50.0
VGG-4	max	68.7	66.8	53.9	56.1	55.0
VGG-4	average	63.7	63.2	52.8	48.9	50.8
VGG-4	attention	67.6	64.6	50.6	46.1	48.3
our CNN	max	68.1	68.7	58.3	55.6	56.9
our CNN	average	68.1	67.3	**59.4**	56.1	57.8
our CNN	attention	**72.6**	**71.8**	58.3	**58.3**	**58.3**

VGG-4 yields to max pooling, the possible reason might be that the larger number of hyper parameters in VGG-4 with attention pooling brings on over fitting. As to the different CNN models, the best results are obtained by our CNN, which means that CNNs with a kernel size of five and no max pooling among convolutional layers appear more suited for this acoustic scene classification task. Our CNN with attention model achieves accuracy of 72.6 % for subtask A, which is a significant improvement over the baseline ($p < .001$ in a one-tailed z-test). In addition, our CNN achieves accuracies of 71.8 %, 58.3 %, and 58.3 % for device A, B, C in subtask B (in a one-tailed z-test, $p < .001$ for device A , $p < .01$ for device B, and $p < .05$ for device C).

The CNN model performs better at subtask A than at subtask B, perhaps because multiple data recording devices were employed for subtask B. The average accuracy of device A is higher than of device B and C in subtask B, which is considered to be caused by the unbalance of data among the three devices; in other words, the dataset contains more data from device A than from device B and C. To investigate our CNN model, a performance comparison of accuracy for each scene class of our best result is presented in Figure 3. Our CNN is optimal for some classes like *park*, *metro station*, and *street traffic*, but it is less able to recognise some classes such as *public square* and *bus*. It is possible that this lower performance is caused by background noise in these classes.

To sum up, our proposed CNNs with an attention model appear helpful to improve the performance over other pooling models for acoustic scene classification tasks.

4. CONCLUSIONS AND PERSPECTIVES

We proposed an attention-based convolutional neural network for acoustic scene classification by building an attention model at the decision level for classification. Based on the official development set of the acoustic scene classification task of the IEEE AASP Challenge on Detection and Classification of Acoustic Scenes and Events (DCASE 2018), our CNN gave better performance than other state-of-the-art CNN models and the proposed attention pooling model performed better than max pooling and average pooling, and achieved a significant improvement of the official baseline at subtask A ($p < .001$ in a one-tailed z-test) and subtask B (in a one-tailed z-test, $p < .001$ for device A , $p < .01$ for device B, and $p < .05$ for device C).

In future works, we will investigate the attention model at the feature level in order to analysis the contributions of feature maps in each convolutional layer. Further, transfer learning will be considered for subtask B, for training a robust model on the dataset from multiple recording sources.

5. ACKNOWLEDGMENT

This work was partially supported by the Horizon H2020 Marie Skłodowska-Curie Actions Initial Training Network European Training Network project under grant agreement No. 766287 (TAPAS), the European Union's Seventh Framework under grant agreement No. 338164 (ERC StG iHEARu), the EPSRC grant EP/N014111/1 "Making Sense of Sounds", and a Research Scholarship from the China Scholarship Council (CSC) No. 201406150082.

6. REFERENCES

[1] B. T. Szabó, S. L. Denham, and I. Winkler, "Computational models of auditory scene analysis: A review," *Frontiers in Neuroscience*, vol. 10, p. 524, Nov. 2016.

[2] Y. Xu, W. J. Li, and K. K. Lee, *Intelligent wearable interfaces*. John Wiley & Sons, 2008.

[3] F. Eyben, F. Weninger, F. Groß, and B. Schuller, "Recent developments in opensmile, the munich open-source multimedia feature extractor," in *Proc. ACM MM*, Barcelona, Catalunya, Spain, 2013, pp. 835–838.

[4] S. Chu, S. Narayanan, C.-C. Kuo, and M. Mataric, "Where am I? Scene recognition for mobile robots using audio features," in *Proc. ICME*, Toronto, Canada, 2006, pp. 885–888.

[5] C. Perera, A. Zaslavsky, P. Christen, and D. Georgakopoulos, "Context aware computing for the internet of things: A survey," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 1, pp. 414–454, May 2013.

[6] E. Marchi, D. Tonelli, X. Xu, F. Ringeval, J. Deng, S. Squartini, and B. Schuller, "Pairwise decomposition with deep neural networks and multiscale kernel subspace learning for acoustic scene classification," in *Proc. DCASE Workshop*, Budapest, Hungary, 2016, pp. 65–69.

[7] K. Qian, Z. Ren, V. Pandit, Z. Yang, Z. Zhang, and B. Schuller, "Wavelets revisited for the classification of acoustic scenes," in *Proc. DCASE Workshop*, Munich, Germany, 2017, pp. 108–112.

[8] M. Chum, A. Habshush, A. Rahman, and C. Sang, "IEEE AASP scene classification challenge using hidden Markov models and frame based classification," in *Proc. DCASE challenge*, Munich, Germany, 2013, 3 pages.

[9] S. Amiriparian, M. Freitag, N. Cummins, and B. Schuller, "Sequence to sequence autoencoders for unsupervised representation learning from audio," in *Proc. DCASE Workshop*, Munich, Germany, 2017, pp. 17–21.

[10] Z. Ren, V. Pandit, K. Qian, Z. Yang, Z. Zhang, and B. Schuller, "Deep sequential image features on acoustic scene classification," in *Proc. DCASE Workshop*, Munich, Germany, 2017, pp. 113–117.

[11] S. H. Bae, I. Choi, and N. S. Kim, "Acoustic scene classification using parallel combination of LSTM and CNN," in *Proc. DCASE Workshop*, Budapest, Hungary, 2016, pp. 11–15.

[12] http://dcase.community/workshop2018/.

[13] W. Zheng, J. Yi, X. Xing, X. Liu, and S. Peng, "Acoustic scene classification using deep convolutional neural network and multiple spectrograms fusion," in *Proc. DCASE Workshop*, Munich, Germany, 2017, pp. 133–137.

[14] S. Adavanne and T. Virtanen, "Sound event detection using weakly labeled dataset with stacked convolutional and recurrent neural network," in *Proc. DCASE Workshop*, Munich, Germany, 2017, pp. 12–16.

[15] Y. Han, J. Park, and K. Lee, "Convolutional neural networks with binaural representations and background subtraction for acoustic scene classification," in *Proc. DCASE Workshop*, Munich, Germany, 2017, pp. 1–5.

[16] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, Apr. 2015.

[17] H. Phan, L. Hertel, M. Maass, and A. Mertins, "Robust audio event recognition with 1-max pooling convolutional neural networks," in *INTERSPEECH*, San Francisco, CA, 2016, pp. 3653–3657.

[18] J. Deng, N. Cummins, J. Han, X. Xu, Z. Ren, V. Pandit, Z. Zhang, and B. Schuller, "The University of Passau open emotion recognition system for the multimodal emotion challenge," in *Proc. CCPR*. Chengdu, China: Springer, 2016, pp. 652–666.

[19] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, "Learning deep features for discriminative localization," in *Proc. CVPR*, Las Vegas, NV, 2016, pp. 2921–2929.

[20] W. Yin, H. Schütze, B. Xiang, and B. Zhou, "ABCNN: Attention-based convolutional neural network for modeling sentence pairs," *Transactions of the Association for Computational Linguistics*, vol. 6, pp. 259–272, Jun. 2016.

[21] K. Chen, J. Wang, L.-C. Chen, H. Gao, W. Xu, and R. Nevatia, "ABC-CNN: An attention based convolutional neural network for visual question answering," *arXiv preprint arXiv:1511.05960*, 2015.

[22] Q. Kong, Y. Xu, W. Wang, and M. D. Plumbley, "Audio Set classification with attention model: A probabilistic perspective," in *Proc. ICASSP*, Calgary, Canada, 2017, no pagination.

[23] J. Foulds and E. Frank, "A review of multi-instance learning assumptions," *The Knowledge Engineering Review*, vol. 25, no. 1, pp. 1–25, Mar. 2010.

[24] Z. Ren, K. Qian, Z. Zhang, V. Pandit, A. Baird, and B. Schuller, "Deep scalogram representations for acoustic scene classification," *IEEE/CAA Journal of Automatica Sinica*, vol. 5, no. 3, pp. 662–669, May 2018.

[25] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. NIPS*, Stateline, NV, 2012, pp. 1097–1105.

[26] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. ICLR*, San Diego, CA, 2015, no pagination.

[27] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *Proc. CVPR 2009*, Miami, FL, 2009, pp. 248–255.

[28] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. ICML*, Lille, France, 2015, pp. 448–456.

[29] M. Simon, E. Rodner, and J. Denzler, "Imagenet pre-trained models with batch normalization," *arXiv preprint arXiv:1612.01452*, 2016.

[30] O. Maron and T. Lozano-Pérez, "A framework for multiple-instance learning," in *Proc. NIPS*, Denver, CO, 1998, pp. 570–576.

[31] J. Amores, "Multiple instance classification: Review, taxonomy and comparative study," *Artificial Intelligence*, vol. 201, pp. 81–105, Aug. 2013.

[32] X. Xu, "Statistical learning in multiple instance problems," Ph.D. dissertation, The University of Waikato, Jun. 2003.

GENERAL-PURPOSE AUDIO TAGGING BY ENSEMBLING CONVOLUTIONAL NEURAL NETWORKS BASED ON MULTIPLE FEATURES

Kevin Wilkinghoff

Fraunhofer Institute for Communication, Information Processing and Ergonomics FKIE
Fraunhoferstraße 20, 53343 Wachtberg, Germany
kevin.wilkinghoff@fkie.fraunhofer.de

ABSTRACT

This paper describes an audio tagging system that participated in Task 2 "General-purpose audio tagging of Freesound content with AudioSet labels" of the "Detection and Classification of Acoustic Scenes and Events (DCASE)" Challenge 2018. The system is an ensemble consisting of five convolutional neural networks based on Mel-frequency Cepstral Coefficients, Perceptual Linear Prediction features, Mel-spectrograms and the raw audio data. For ensembling all models, score-based fusion via Logistic Regression is performed with another neural network. In experimental evaluations, it is shown that ensembling the models significantly improves upon the performances obtained with the individual models. As a final result, the system achieved a Mean Average Precision with Cutoff 3 of 0.9414 on the private leaderboard of the challenge.

***Index Terms*—** audio tagging, acoustic event classification, convolutional neural network, score-based fusion

1. INTRODUCTION

In past years, deep convolutional neural networks (CNN) or variants thereof have taken over almost any area related to image classification and computer vision in general. As audio data can easily be converted into images by computing their spectrogram, they have also become popular for acoustic event detection and classification [1, 2, 3]. Moreover, CNNs trained on spectrograms are reported to outperform classical approaches as for example Hidden Markov Models (HMMs) [4]. In this work, an ensemble of CNNs for the purpose of acoustic event classification is presented. The ensemble consists of five different CNNs trained on multiple features derived from the acoustic data and another neural network used for fusing the scores. More concretely, Mel-frequency Cepstral Coefficients (MFCCs) [5], Perceptual Linear Prediction (PLP) features [6], Mel-spectrograms and the raw data are used as features.

The presented system participated in Task 2 "General-purpose audio tagging of Freesound content with AudioSet labels" of the "Detection and Classification of Acoustic Scenes and Events (DCASE)" challenge 2018. The audio dataset being used consists of a subset of the Freesound dataset (FSD) [7] which has been taken from the online database Freesound [8]. All files in FSD have been uploaded and tagged by many different users. A mapping from these tags to AudioSet [9] categories has been manually designed and all samples have been automatically annotated. After that, the annotations of some files have been manually verified although there are some cases where the annotators could not agree on a label.

As a result, the part of the dataset used for training consists of 9473 files belonging to one of 41 categories. The distribution is not uniform but ranges from 94 to 300 samples per category. In addition to that, the length of the audio files is between 300 ms and 30 s. Only 3710 samples are manually verified and for each file it is known if this is the case or not. According to the organizers of the challenge, at least two thirds of the non-verified samples are estimated to be labeled correctly for each category. The test set consists of 1600 manually verified samples and 7800 non-verified samples referred to as padding files. Furthermore, 301 files have been used as validation data for the public leaderboard and 1299 files have been used for the final evaluation of the challenge (private leaderboard). The padding files have not been used for any evaluation and were simply kept as test data to prevent participants from cheating. For a more detailed description of the challenge and dataset the reader is referred to [10].

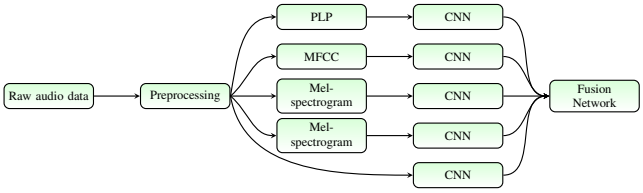


Figure 1: Structure of the audio tagging system

This paper is organized as follows. First, the global structure of the audio tagging system is presented. Then, all five CNNs based on all features including their hyperparameters as well as the Fusion Network are described. After that, experimental results are given in order to compare the performance of the CNNs and to show that creating an ensemble of them is highly beneficial.

2. STRUCTURE OF THE AUDIO TAGGING SYSTEM

An overview of the audio tagging system's structure is depicted in Fig. 1. The system is an ensemble consisting of five different CNNs trained on PLP features, MFCCs, Mel-spectrograms and the raw audio data. In the following, we will refer to all of them as features. First, the audio data has been preprocessed with Librosa [11]. All files have been downsampled from 44100 Hz to 24000 Hz and were normalized with respect to the maximum norm. Any part of the files that is 50 db below peak power is considered as silence and has been removed. After these steps, all features have been extracted with Librosa except the PLP features which were computed via Sidekit [12]. We used a dimension of size 22 for the PLP features, one of size 64 for the MFCC features and one of size 96 as well as 128 for the Mel-spectrograms.

Since CNNs need input data of fixed size and the audio files vary greatly in length, we partitioned the sequence of features into

overlapping windows of length 500 with an overlap of 50 (ordered with respect to time). To enforce a more different behavior of both CNNs trained on Mel-spectrograms, a window size of 200 with overlap 20 has been used for the 128 dimensional features instead. Much larger windows of length 48000 (i.e. clips of 2 seconds) with an overlap of 4800 have been used for the raw data. In case that any of the feature windows was too short, the window has been repeated until the desired length was reached. Later, the geometric mean of the scores obtained with all windows belonging to the same file has been taken as the final score for that file. Note that a similar windowing approach is also used for the baseline system of the challenge presented in [10].

The output scores of all CNNs obtained with the validation data are fused with a neural network. It basically applies Logistic Regression (see [13]) with only a single parameter per model and one global bias. Thus, the final scores, used for identifying the classes by returning the argument of the maximum score, are computed as a weighted linear combination of all models' scores. This has the advantage that better performing models have a higher influence than weaker ones and usually leads to a better performance than taking the mean (see [14]). A weighted geometric mean of the probabilities corresponds to a weighted sum (i.e. a standard linear transformation of a neural network) in log-space. Therefore, we converted all softmax probabilities to log-space before applying the neural network as this improved the overall performance of the system. Additionally, we standardized the scores of each model i.e. we subtracted the mean and divided by the standard deviation of all scores.

All hyperparameters involved have been tuned using Stratified 5-Fold Cross Validation as implemented in Scikit-learn [15]. After training all models on the five data splits, also five ensembles were obtained and each of them has been applied to the test dataset instead of retraining each model on the full training dataset. Hence, the full ensemble consists of 25 submodels. To have a single final output, we simply took the geometric mean of the fused output probabilities obtained by evaluating the five model ensemble with the test dataset. By doing so, the whole training dataset has been used while also having the possibility to monitor the system's performance using the validation data. Thus, we could apply Early Stopping when training each CNN by monitoring the validation loss. Furthermore, the scores obtained with the validation data could be used to train the neural networks used for fusing the scores which would not have been possible otherwise. As a result, a slightly better performance was reached.

3. NEURAL NETWORK ARCHITECTURES

All CNNs and the Fusion Network have been implemented using Keras [16] with Tensorflow [17]. Their structures are depicted in Tab. 1, 2, 3, 4, 5 and 6 which can be found in the appendix. To be able to compare the complexities of all models, we included their total number of parameters in Tab. 7.

As stated in [2, 18], data augmentation is of great help in order to have a well performing system. Therefore, Mix-up [19] with $\alpha = 1$, Cutout [20], Dropout [21] and vertical shifts up to 60% of the total width have been applied randomly in each batch. When using the Mel-spectrogram based CNNs, we also used feature-wise centering as well as featurewise normalization of the standard deviation. All CNNs have been trained for 800 epochs with a batch size of 64 by minimizing the Categorical Crossentropy. To speed up the training process, Adam [22] with a learning rate of 0.001 and weight decay of 0.0001 as well as Batch-Normalization [23] have been used. Due to the non-uniform distribution of training samples per class, we used balanced class weights during training to put more emphasis on the classes having fewer samples as implemented in Scikit-learn (which is loosely following [24]).

As noted above, many labels are not guaranteed to be correct due to the automatic labeling procedure. To compensate for this, Label Smoothing [25] has been applied. This means that categorical labels are altered by reducing the single 1 to $1 - \alpha_{nv}$ and replacing all zeros with $\frac{\alpha_{nv}}{N-1}$ where $\alpha_{nv} \in [0, 1]$ and $N \in \mathbb{N}$ denotes the number of classes. According to the organizers of the challenge, at least two thirds of the non-verified labels are correct. Because of that, a value of $\alpha_{nv} = 0.3$ has been used for all non-verified samples. As the annotators could not agree on a label for some of the files and thus at least some labels are probably wrong, we used a value of $\alpha_v = 0.05$ for the manually verified data. In addition to that, sounds recorded in realistic environments, naturally, contain more than one class although the organizers of the challenge tried to prevent this. For example, a barking dog could have been recorded outside in the streets where buses or other vehicles are also present or a snare drum, base drum and hi-hat are played together in one recording of a drummer. Thus, it is not preferable that the system tries to make a hard prediction per file which is another reason to include Label Smoothing and Mix-up.

For the Fusion Network slightly different parameters and no data augmentation techniques have been used. It has been trained for 1000 epochs with a batch size equal to the size of the full dataset (i.e. with the exact gradient) using Adam with a learning rate of 0.01. Instead of Label Smoothing and Weight Decay, L2 Regularization with a penalty of 0.0005 has been applied. When using batch normalization, it was important to disable the additional beta and gamma parameters in Keras. Otherwise, there is an additional linear transformation with independent weights for each class and each model. This gives the Fusion Network too much power and the generalization to unseen test data is much worse.

4. EXPERIMENTAL RESULTS

The mean Average Precisions with Cutoff 3 (mAP@3s) [8], we obtained in the challenge, can be found in Fig. 2 and have been depicted for each model individually. They relate to the dataset used for the public leaderboard and the private leaderboard, which correspond to the final challenge results, as well as our own five validation splits of the training dataset. To make it more clear, since all five validation sets of the splits together result in the whole training set again, each mAP@3 is based on all 9473 training files. Note that the data split used for the public leaderboard only consists of 301 files and thus is a less accurate estimate of the true mAP@3.

As expected, the obtained mAP@3s are much higher when only evaluating with the manually verified data but still using the non-verified samples for training. Thus, the assumption that the weakly-labeled files are by far not optimally labeled seems to be true. Since complete the test set is manually verified, the results of the challenge are closer to the presented mAP@3s of the manually verified subset than to the full validation set. It is also clear that the performance obtained with the validation data is a bit too optimistic because all parameters have been adapted to these data splits. The CNNs trained on Mel-spectrograms perform better than the ones trained on PLP and MFCC features which both result in similar mAP@3s. In addition to that, the CNN trained on the raw data performed worst but still reasonably well. Interestingly, a relatively small number of model parameters seems to be sufficient since both

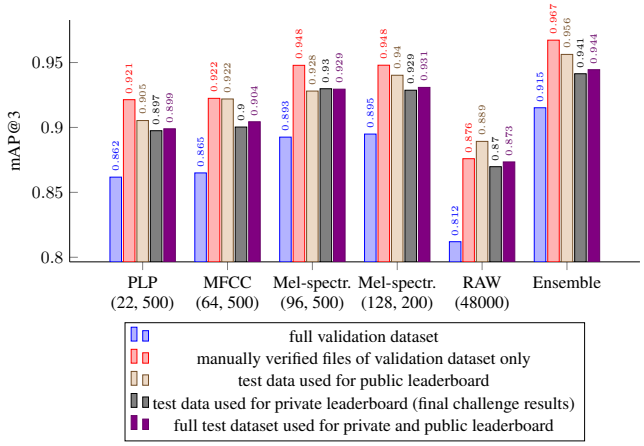


Figure 2: Mean Average Precision with Cutoff 3 obtained with the CNNs and the ensemble.

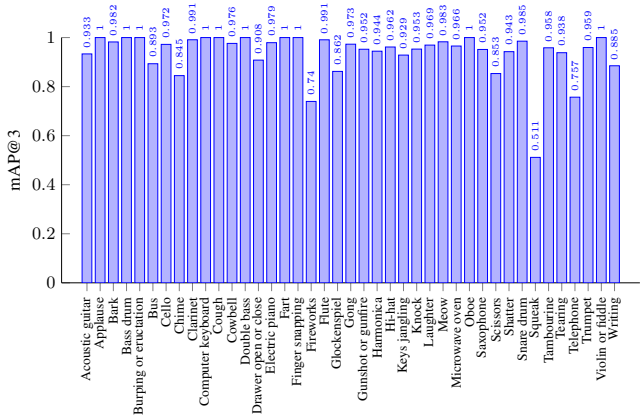


Figure 3: Mean Average Precision with Cutoff 3 per class obtained with all test files used for the public and private leaderboard.

Mel-spectrogram based CNNs performed equally well although one has about five times as many parameters as the other (see Tab. 7). Furthermore, it can be seen that the ensemble of all CNNs has a significantly higher mAP@3 than all of its components alone across all datasets. Hence, each of them has a slightly different view on the data providing additional information. In conclusion, building an ensemble of CNNs based on all the features is highly beneficial. But compared to the baseline system which yields an mAP@3 of 0.6945 on the private leaderboard and of 0.7049 on the public one, even the individual CNNs perform much better.

Next, the mAP@3s have been evaluated for each class individually to see which classes are easily identified and which ones are not. See Fig. 3, for a visualization. The mAP@3s for most classes are decent and ten classes are even identified without any errors. However, there are some classes whose mAP@3s are much lower than those of the other classes. The worst performing class is "Squeak" which has an mAP@3 of 0.511. A reason may be that squeaks are diversely sounding depending on the objects causing them. Looking at the confusion matrix (see Fig. 4), it can be seen that squeaks are fairly often confused with many other classes which supports this assumption. Additionally, there are also classes which are almost exclusively confused with each other as for example "Chime"

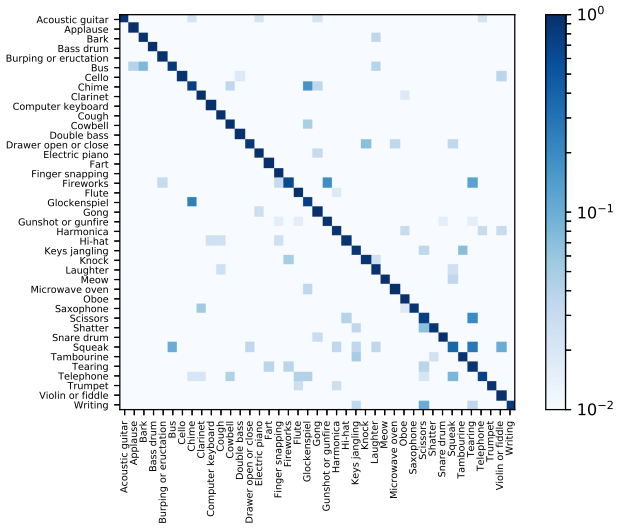


Figure 4: Confusion matrix obtained with all test files used for the public and private leaderboard.

and "Glockenspiel". Both have a relatively low mAP@3 of approximately 0.85. From a human's perspective it seems to be reasonable to confuse them as they are mostly perceived as similarly sounding.

5. CONCLUSIONS AND FUTURE WORK

In this work, an audio tagging system consisting of five CNNs and a neural network for fusing their scores has been presented and described. In the experimental evaluations conducted for Task 2 of the DCASE Challenge 2018, it has been shown that ensembling all models significantly improves upon the performances of each individual model. Furthermore, we examined the performance per class and were able to identify classes which are very easy to classify and some that are difficult to classify leading to more insights about the dataset used in this challenge.

An improvement of the presented audio tagging system could be achieved by using additional data augmentation techniques. Some techniques to be named are Pitch-Shifting, Time-Stretching, class conditional data augmentation as suggested in [18] or Equalized Mixture Data Augmentation [2] which is an extension of Mix-up. Another possibility is to make use of Transfer Learning and replace some models of the ensemble with pretrained deep neural networks as for example VGG or ResNet (as done in [3]). This will probably improve the performances of the individual models and thus also the capabilities of the whole audio tagging system. Last but not least, another improvement may be accomplished by designing custom features or classifiers (e.g. other CNNs) for differentiating among those classes which are hard to tell apart and add them to the ensemble.

6. ACKNOWLEDGMENT

The author would like to thank Daisuke Niizumi who also participated in this challenge and shared his insights publicly in the Kaggle discussion forum. His comments on data augmentation techniques such as Mix-up and Cutout have been of great help to improve the performance of the presented audio tagging system.

7. REFERENCES

[1] O. Gencoglu, T. Virtanen, and H. Huttunen, "Recognition of acoustic events using deep neural networks," in *Signal Processing Conference (EUSIPCO), 2014 Proceedings of the 22nd European*. IEEE, 2014, pp. 506–510.

[2] N. Takahashi, M. Gygli, B. Pfister, and L. Van Gool, "Deep convolutional neural networks and data augmentation for acoustic event detection," in *Interspeech*, 2016, pp. 2982–2986.

[3] S. Hershey, S. Chaudhuri, D. P. Ellis, J. F. Gemmeke, A. Jansen, R. C. Moore, M. Plakal, D. Platt, R. A. Saurous, B. Seybold, *et al.*, "Cnn architectures for large-scale audio classification," in *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*. IEEE, 2017, pp. 131–135.

[4] A. Mesaros, T. Heittola, A. Eronen, and T. Virtanen, "Acoustic event detection in real life recordings," in *Signal Processing Conference (EUSIPCO), 2010 Proceedings of the 18th European*. IEEE, 2010, pp. 1267–1271.

[5] S. Davis and P. Mermelstein, "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences," *IEEE transactions on acoustics, speech, and signal processing*, vol. 28, no. 4, pp. 357–366, 1980.

[6] H. Hermansky, "Perceptual linear prediction (plp) analysis of speech," *the Journal of the Acoustical Society of America*, vol. 87, no. 4, pp. 1738–1752, 1990.

[7] E. Fonseca, J. Pons, X. Favory, F. Font, D. Bogdanov, A. Ferraro, S. Oramas, A. Porter, and X. Serra, "Freesound datasets: a platform for the creation of open audio datasets," in *Proceedings of the 18th International Society for Music Information Retrieval Conference (ISMIR 2017)*, Suzhou, China, 2017, pp. 486–493.

[8] F. Font, G. Roma, and X. Serra, "Freesound technical demo," in *Proceedings of the 21st ACM international conference on Multimedia*. ACM, 2013, pp. 411–412.

[9] J. F. Gemmeke, D. P. W. Ellis, D. Freedman, A. Jansen, W. Lawrence, R. C. Moore, M. Plakal, and M. Ritter, "Audio set: An ontology and human-labeled dataset for audio events," in *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*. IEEE, 2017, pp. 776–780.

[10] E. Fonseca, M. Plakal, F. Font, D. P. W. Ellis, X. Favory, J. Pons, and X. Serra, "General-purpose tagging of freesound audio with audioset labels: Task description, dataset, and baseline," *Submitted to DCASE2018 Workshop, 2018. URL: https://arxiv.org/abs/1807.09902*.

[11] B. McFee, C. Raffel, D. Liang, D. P. Ellis, M. McVicar, E. Battenberg, and O. Nieto, "librosa: Audio and music signal analysis in python," in *Proceedings of the 14th python in science conference*, 2015, pp. 18–25.

[12] A. Larcher, K. A. Lee, and S. Meignier, "An extensible speaker identification sidekit in python," in *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*. IEEE, 2016, pp. 5095–5099.

[13] C. M. Bishop, *Pattern Recognition and Machine Learning*, 1st ed. Springer, 2006.

[14] K. Wilkinghoff, P. M. Baggenstoss, A. Cornaggia-Urrigshardt, and F. Kurth, "Robust speaker identification by fusing classification scores with a neural network," 2018, preprint, to appear in: Proceedings of the 13th ITG Symposium on Speech Communication.

[15] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[16] F. Chollet *et al.*, "Keras," https://keras.io, 2015.

[17] M. Abadi *et al.*, "Tensorflow: a system for large-scale machine learning," *OSDI*, vol. 16, pp. 265–283, 2016.

[18] J. Salamon and J. P. Bello, "Deep convolutional neural networks and data augmentation for environmental sound classification," *IEEE Signal Processing Letters*, vol. 24, no. 3, pp. 279–283, 2017.

[19] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," *arXiv preprint arXiv:1710.09412*, 2017.

[20] T. DeVries and G. W. Taylor, "Improved regularization of convolutional neural networks with cutout," *arXiv preprint arXiv:1708.04552*, 2017.

[21] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[22] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[23] S. Ioffe and C. Szegedy, "Batch normalization: accelerating deep network training by reducing internal covariate shift," in *Proceedings of the 32nd International Conference on Machine Learning*, vol. 37, 2015, pp. 448–456.

[24] G. King and L. Zeng, "Logistic regression in rare events data," *Political analysis*, vol. 9, no. 2, pp. 137–163, 2001.

[25] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2818–2826.

Table 1: Architecture of the PLP based CNN.

Layer	Output Shape	#Parameters
Input	(22, 500)	0
Convolution (kernel size: 5x5, strides: 2x3, ReLU)	(11, 167, 64)	1,664
Max-Pooling (pool size: 2x2, strides: 1x2)	(10, 83, 64)	0
Batch Normalization	(10, 83, 64)	256
Convolution (kernel size: 5x5, ReLU)	(10, 83, 128)	204,928
Max-Pooling (pool size: 2x2, strides: 1x2)	(9, 41, 128)	0
Batch Normalization	(9, 41, 128)	512
Convolution (kernel size: 3x3, ReLU)	(9, 41, 192)	221,376
Batch Normalization	(9, 41, 192)	768
Convolution (kernel size: 3x3, ReLU)	(9, 41, 256)	442,624
Batch Normalization	(9, 41, 256)	1,024
Convolution (kernel size: 3x3, ReLU)	(9, 41, 256)	262,400
Max-Pooling (pool size: 2x2, strides: 1x2)	(8, 20, 256)	0
Batch Normalization	(8, 20, 256)	1,024
Flatten	40,960	0
Dropout (0.5)	40,960	0
Dense (ReLU)	256	10,486,016
Batch Normalization	256	1,024
Dropout (0.5)	256	0
Dense (ReLU)	128	32,896
Batch Normalization	128	512
Dense (Softmax)	41	5,289

Table 2: Architecture of the MFCC based CNN.

Layer	Output Shape	#Parameters
Input	(64, 500)	0
Convolution (kernel size: 5x5, strides: 2x3, ReLU)	(32, 167, 64)	1,664
Max-Pooling (pool size: 2x2, strides: 1x2)	(31, 83, 64)	0
Batch Normalization	(31, 83, 64)	256
Convolution (kernel size: 5x5, ReLU)	(31, 83, 128)	204,928
Max-Pooling (pool size: 2x2, strides: 1x2)	(30, 41, 128)	0
Batch Normalization	(30, 41, 128)	512
Convolution (kernel size: 3x3, ReLU)	(30, 41, 192)	221,376
Batch Normalization	(30, 41, 192)	768
Convolution (kernel size: 3x3, ReLU)	(30, 41, 256)	442,624
Batch Normalization	(30, 41, 256)	1,024
Convolution (kernel size: 3x3, ReLU)	(30, 41, 256)	262,400
Max-Pooling (pool size: 2x2, strides: 1x2)	(29, 20, 256)	0
Batch Normalization	(29, 20, 256)	1,024
Flatten	148,480	0
Dropout (0.5)	148,480	0
Dense (ReLU)	256	38,011,136
Batch Normalization	256	1,024
Dropout (0.5)	256	0
Dense (ReLU)	128	32,896
Batch Normalization	128	512
Dense (Softmax)	41	5,289

Table 3: Architecture of the Mel Spectrogram based CNN.

Layer	Output Shape	#Parameters
Input	(96, 500)	0
Convolution (kernel size: 5x5, strides: 2x3, ReLU)	(48, 167, 64)	1,664
Max-Pooling (pool size: 3x3, strides: 1x2)	(46, 83, 64)	0
Batch Normalization	(46, 83, 64)	256
Convolution (kernel size: 5x5, ReLU)	(46, 83, 128)	204,928
Max-Pooling (pool size: 3x3, strides: 2x2)	(22, 41, 128)	0
Batch Normalization	(22, 41, 128)	512
Convolution (kernel size: 3x3, ReLU)	(22, 41, 192)	221,376
Batch Normalization	(22, 41, 192)	768
Convolution (kernel size: 3x3, ReLU)	(22, 41, 256)	442,624
Batch Normalization	(22, 41, 256)	1,024
Convolution (kernel size: 2x2, ReLU)	(22, 41, 256)	262,400
Max-Pooling (pool size: 2x2)	(11, 20, 256)	0
Batch Normalization	(11, 20, 256)	1,024
Flatten	56,320	0
Dropout (0.5)	56,320	0
Dense (ReLU)	256	14,418,176
Batch Normalization	256	1,024
Dropout (0.5)	256	0
Dense (ReLU)	128	32,896
Batch Normalization	128	512
Dense (Softmax)	41	5,289

Table 4: Architecture of the Mel Spectrogram based CNN.

Layer	Output Shape	#Parameters
Input	(128, 200)	0
Convolution (kernel size: 5x5, strides: 2x3, ReLU)	(64, 67, 64)	1,664
Max-Pooling (pool size: 3x3, strides: 1x2)	(62, 33, 64)	0
Batch Normalization	(62, 33, 64)	256
Convolution (kernel size: 5x5, ReLU)	(62, 33, 128)	204,928
Max-Pooling (pool size: 3x3, strides: 2x2)	(30, 16, 128)	0
Batch Normalization	(30, 16, 128)	512
Convolution (kernel size: 3x3, ReLU)	(30, 16, 192)	221,376
Batch Normalization	(30, 16, 192)	768
Convolution (kernel size: 3x3, ReLU)	(30, 16, 256)	442,624
Max-Pooling (pool size: 2x2)	(15, 8, 256)	0
Batch Normalization	(15, 8, 256)	1,024
Convolution (kernel size: 3x3, ReLU)	(15, 8, 256)	590,080
Max-Pooling (pool size: 2x2)	(7, 4, 256)	0
Batch Normalization	(7, 4, 256)	1,024
Convolution (kernel size: 3x3, ReLU)	(7, 4, 512)	1,180,160
Batch Normalization	(7, 4, 512)	2,048
Global Average Pooling	512	0
Dropout (0.5)	56,320	0
Dense (ReLU)	256	131,328
Batch Normalization	256	1,024
Dropout (0.5)	256	0
Dense (ReLU)	128	32,896
Batch Normalization	128	512
Dense (Softmax)	41	5,289

Table 5: Architecture of the raw data based CNN.

Layer	Output Shape	#Parameters
Input	48000	0
Convolution (kernel size: 31, strides: 4, ReLU)	(12000, 64)	2,048
Batch Normalization	(12000, 64)	256
Max-Pooling (pool size: 16)	(750, 64)	0
Dropout (0.2)	(750, 64)	0
Convolution (kernel size: 15, strides: 2, ReLU)	(375, 128)	123,008
Batch Normalization	(375, 128)	512
Max-Pooling (pool size: 4)	(93, 128)	0
Dropout (0.2)	(93, 128)	0
Convolution (kernel size: 8, ReLU)	(93, 192)	196,800
Batch Normalization	(93, 192)	768
Dropout (0.2)	(93, 192)	0
Convolution (kernel size: 4, ReLU)	(93, 256)	196,864
Batch Normalization	(93, 256)	1,024
Global Average Pooling	256	0
Dropout (0.5)	256	0
Dense (ReLU)	256	65,792
Batch Normalization	256	1,024
Dropout (0.5)	256	0
Dense (ReLU)	128	32,896
Batch Normalization	128	512
Dense (Softmax)	41	5,289

Table 6: Architecture of the Fusion Network.

Layer	Output Shape	#Parameters
Input	205	0
Batch Normalization	205	410
Fusion Layer (Softmax)	41	6

Table 7: Total number of parameters of each neural network.

Model	Input size	Total #Parameters
MFCC based CNN	(64, 500)	39,184,873
PLP based CNN	(22, 500)	11,659,753
Mel-spectrogram based CNN	(96, 500)	15,591,913
Mel-spectrogram based CNN	(128, 200)	2,817,513
raw data based CNN	48000	624,745
Fusion Network	205	416

A REPORT ON AUDIO TAGGING WITH DEEPER CNN, 1D-CONVNET AND 2D-CONVNET

Qingkai WEI, Yanfang LIU, Xiaohui RUAN

Beijing Kuaiyu Electronics Co., Ltd., Beijing, PRC.
{wqk, liuyf, rxh}@kuaiyu.com

ABSTRACT

General-purpose audio tagging is a newly proposed task in DCASE 2018, which can provide insight towards broadly-applicable sound event classifiers. In this paper, two systems (named as 1D-ConvNet and 2D-ConvNet in this paper) with small kernel sizes, multiple functional modules, deeper CNN (convolutional neural networks) are developed to improve performance in this task. In detail, different audio features are used, i.e. raw waveforms are for 1D-ConvNet, while frequency domain features, such as mfcc, log-mel spectrogram, multi-resolution log-mel spectrogram and spectrogram, are utilized as the 2D-ConvNet input. Using DCASE 2018 Challenge task 2 dataset to train and evaluate, the best single model with 1D-ConvNet and 2D-ConvNet are chosen, whose kaggle public leaderboard score are 0.877 and 0.961 respectively. In addition, a better ensemble rank averaging prediction get a score 0.967 on the public leaderboard, ranking 5/558, while score 0.942 on the private leaderboard ranking 11/558.

Index Terms— DCASE 2018, Audio tagging, Convolutional neural networks, 1D-ConvNet, 2D-ConvNet, Model ensemble

1. INTRODUCTION

In recent years, computer vision techniques such as object detection and segment, are applied in monitoring, surveillance and autonomous driving. In the process of these techniques' performance improved from laboratory to applications, development of neural network architectures played an important role. Along with the appearance of LeNet [1], Alexnet [2], VGG Net[3], GoogLeNet (Inception V1 and following V3, V4) [4, 5, 6], Deep Residual Net [7], Squeeze-and excitation networks [8], neural networks become much deeper, together with the ingenious modules such an inception modules, factorizing convolutions, residual blocks and so on.

Similar to vision, audio also takes lots of unique information, which can help people recognize their surroundings together with vision or tactile information. However, corresponding techniques such as sound event detection and specific sound extraction have not been brought to general applications.

Sound event detection is a system to automatically detect and classify emergency sound events. In 1st DCASE challenge (DCASE 2013, IEEE AASP Challenge: Detection and Classification of Acoustic Scenes and Events), sound event detection was firstly focused together with audio scene classification [9]. Then in DCASE 2016 challenge, audio tagging was introduced as a new task. Audio tagging aims at putting one or several sound events tags on a sound clip, like "domestic", "musical instruments", "animals", "human sounds", "speech". This task can provide insight to broadly-applicable sound event classifiers, with increasing amount of sound event categories. And it can be applied in areas such as audio surveillance [10], information retrieval [11], automatic description of multimedia.

Since 2013, the algorithms on audio tagging and sound event detection have been mainly shifted from traditional classifier approaches (mfcc-gmm, HMM: hidden Markov model, NMF: nonnegative matrix factorization, random forests) [9, 12] to deep learning methods such as DNN [13, 14, 15], CNN [16, 17], RNN [18].

As to audio features, many frequency domain features such as mfcc (mel-frequency cepstrum coefficients) [15], mel-spectrogram [13] and spectrogram [19] have been used in similar tasks. Moreover, raw waveform is also applied as the input to classifiers in some recent work about acoustic scene recognition and speech recognition [19, 20, 21].

However, there is no universally accepted conclusion about which neural network and audio feature are best. In this audio tagging task, inspired by the process of neural network evolutions in computer vision, we applied two deeper convolutional neural networks (1D-ConvNet with raw waveforms as input, 2D-ConvNet with frequency domain features as input) to improve the performance. Several techniques which work well in computer vision are applied effectively in this audio tagging task:

- The neural network architectures are much deeper (1D-ConvNet 18 layers, 2D-ConvNet 32 layers), with inception modules, factorizing convolutions, residual blocks applied, which lead to much better performance;
- For 2D-ConvNet, different frequency domain audio features are compared with the same model, including mfcc, log-mel spectrogram, multi-resolution log-mel spectrogram and spectrogram;
- Data augmentation methods such as mixup, random erase are used, which are effective to overcome overfitting;
- Model ensemble techniques are used, predictions of 1D-ConvNet and 2D-ConvNet are combined with rank averaging method. More model ensemble techniques like stacking should be tested in future;
- Training and validation based on DCASE 2018 task 2 dataset verify the effectiveness of the proposed methods.

The rest of this paper is organized as follows. Section 2 describes the features, data augmentation methods, architectures and parameters of these two neural networks. Section 3 shows the experiment setup and performances with DCASE 2018 task2 dataset. Submissions and conclusions are presented in Section 4.

http://www.kuaiyu.com/en, the leading enterprise in audio security monitoring industry in China.

2. METHODS

The architectures of two neural networks, 1D-ConvNet and 2D-ConvNet, are shown in Table. 1 and 2. For 1D-ConvNet, raw waveforms with normalization are set as input directly. While for 2D-ConvNet, the features including mfcc, log-mel spectrogram, multi-resolution log-mel spectrogram, spectrogram are extracted from raw waveforms. The output of the neural network is the probabilities of 41 classes, between 0 and 1, with sum as 1. Details about feature extraction, data augmentation and neural networks are described then.

2.1. Features and data augmentation

For 1D-ConvNet, the raw time-domain waveforms are directly used as input at 44100 Hz. The original data length of train and test samples are range from 300 ms to 30 s. To get input for 1D-ConvNet, waveforms of a few seconds are randomly (with random offset) extracted from the raw waveforms. The length of extracted waveforms are set as 2s, 3s, 4s, 5s, to compare the performances in this task. It should be noticed that longer extracted waveforms would lead to more computationally expensive resources.

For 2D-ConvNet, we study the performances of different frequency domain features. The features selected are mfcc, log-mel spectrogram, multi-resolution log-mel spectrogram [22] and spectrogram. The basic parameters are same: sample frequency 44100 Hz, window size 2048 samples (46.44 ms), hop size 512 samples (11.61 ms) with pre-fft Hamming window. As it demonstrated above, same length of waveforms are randomly extracted from raw data firstly, transformed to T frames. For four different features, other parameters are list below:

mfcc:

Number of mfccs 40, feature size $T \times 40$.

log-mel spectrogram:

Number of mel filters 128, feature size $T \times 128$.

multi-resolution log-mel spectrogram:

It's concluded that log mel-band energy extracted in multi-resolution windows give considerable improvement [22]. We wish to examine its effect with deeper CNN, so the window sizes are 2048, 8192 and 16384 samples, with feature size $T \times (128 * 3)$ shown as Fig. 1

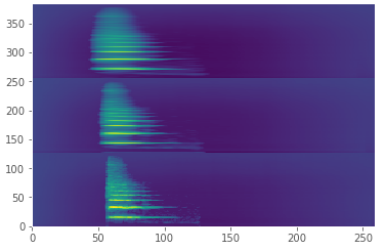


Figure 1: Example of multi-resolution log-mel spectrogram feature.

spectrogram:

Number of frequency bins is 513, feature size $T \times 513$.

Data augmentation methods such as mixup [23] and random erasing [24], are applied to the frequency domain features, which help eliminating overfitting effectively. Preprocessing methods like silence trim are also examined, which did not show improvement of performance.

2.2. Neural networks

1D-ConvNet (parameters: 2,099,801)
Input: 44100·t 1D time-domain waveform
conv1d, kernel 80, stride 4, 48
max pool, 4, stride 4
$\begin{bmatrix}\text{conv1d, kernel 3, stride 1, 48}\\ \text{conv1d, kernel 3, stride 1, 48}\end{bmatrix} \times 2$
max pool, 4, stride 4
$\begin{bmatrix}\text{conv1d, kernel 3, stride 1, 96}\\ \text{conv1d, kernel 3, stride 1, 96}\end{bmatrix} \times 2$
max pool, 4, stride 4
$\begin{bmatrix}\text{conv1d, kernel 3, stride 1, 192}\\ \text{conv1d, kernel 3, stride 1, 192}\end{bmatrix} \times 2$
max pool, 4, stride 4
$\begin{bmatrix}\text{conv1d, kernel 3, stride 1, 384}\\ \text{conv1d, kernel 3, stride 1, 384}\end{bmatrix} \times 2$
Global average pooling (output: 41)
Softmax

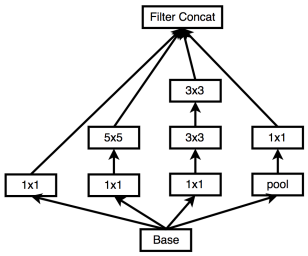
Table 1: Architectures of 1D-ConvNet with time-domain waveform inputs [21]. $[...] \times k$ denotes the k stacked layers. Double layers in a bracket denotes a residual block [7]. Convolutional layers are followed with BN and ReLU, which are not shown in the table.

1D-ConvNet takes time-domain waveforms as input, which are represented as a long 1D vector. The neural network is same as that in the paper [21], details are shown as Table. 1. For t seconds long waveforms, the input layer is a 44100·t 1D vector. To build this deep CNN, small kernel sizes are used for convolutional layers. Basic modules like batch normalization, rectified linear units are applied following each convolutional layer. Network depth is very important to get better accuracy. However, with the depth of network increasing, accuracy can get saturated and degrade. To construct effective deeper network, residual blocks can help a lot [7]. In 1D-ConvNet, two convolutional layers in a bracket denotes a residual block.

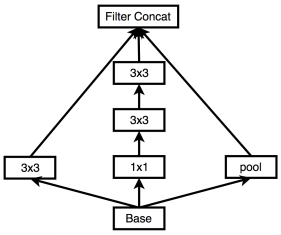
2D-ConvNet (parameters: 7,664,969)
Input: $299 \times 299 \times 3$ frequency-domain features
conv2d, kernel 3×3, stride 2, 32
conv2d, kernel 3×3, stride 1, 32
conv2d, kernel 3×3, stride 1, 64
max pool, 3, stride 2
[inception block A as Fig. 2(a)] $\times 3$
[inception block B as Fig. 2(b)] $\times 1$
[inception block C as Fig. 2(c)] $\times 3$
Global average pooling
Dense 1024 (output: 41)
Softmax

Table 2: Architectures of 2D-ConvNet network for frequency-domain features. $[...] \times k$ denotes the k stacked layers. Details of inception blocks can be seen in Fig. 2. Convolutional layers are followed with BN and ReLU, which are not shown in the table.

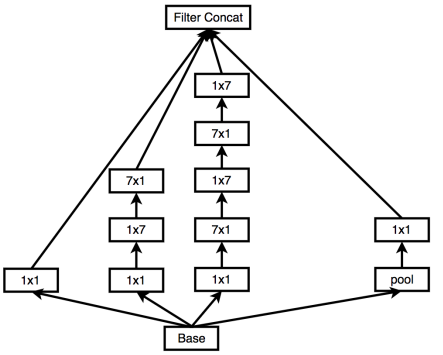
For 2D-ConvNet, frequency domain audio features are used as input. As Sec. 2.1 described, the features' size can be $T \times 40$, $T \times 128$, $T \times (128 * 3)$ or $T \times 513$. Here, T is set as 299, about



(a) Inception block A.



(b) Inception block B.



(c) Inception block C.

Figure 2: Details of inception blocks.

3.5 s. To match the neural network, features are resized to the shape $(299, 299, 3)$. The input size is 3 channel, when we accidentally use 3 channel input, the score increased a lot than that of 1 channel input. The may because 3 channel neural network has better ability to represent. As it concluded in [5], inception modules can widen the network with multiple sizes of kernel in the same layer and factorizing convolutions decrease parameters a lot. They are applied in 2D-ConvNet, played an important role in the improvement of performance. Details are shown in Table. 2.

3. SETUP AND PERFORMANCE EVALUATION

3.1. Dataset and evaluation metric

DCASE 2018 task 2 dataset is used to train and evaluate above two neural networks. Categories of sound event include musical instruments, human sounds, domestic sounds, animals. Recording scenarios and techniques can be very different as sounds are uploaded by users all around the world. The labeling of the samples is a mapping from Freesound tags to AudioSet Ontology categories, which may not so match with the content of samples [25, 26]. The train set includes 9473 samples while the number of audio samples per category ranges from 94 to 300. 3710 of 9473 annotations of samples is manually verified while the others are not. The test set includes 9400 samples, with about 1.6k manually-verified annotations with a similar category distribution, used for evaluating the system.

We tried to do manually-verify to the rest of train set, and used verified labels to train 2D-ConvNet. However, the score decrease from above 0.95 to below 0.70, which shows a much worse performance. So for the final submissions, the original training labels are used. When doing the manually verify, we found several tips that make this task difficult:

- Some categories are really hard to classify even by people, for example (Chime, Cowbell, Glockenspiel) or (Flute, Clarinet);
- With below 300 samples, some categories can be fully representative, e.g. most samples of 'Laugher' is a 'evil' type in train set;
- Some samples can be with multi-label.

To evaluate each developed system, predictions should be uploaded to kaggle platform and are evaluated with the Mean Average Precision @ 3 metric. The kaggle platform can give a public leaderboard score with approximately 19% of the test data (about 300 samples). The final results are based on the rest 81%. We ever worried about that if public and private test data are independent identically distributed, while public test data have about 300 samples of 41 categories. The final leaderboard shows that most participants' predictions are overfitting.

3.2. Baseline

The baseline method is provided, giving a sense of performance possible with the above dataset. The baseline system implements a CNN classifier, with frames of log-mel spectrogram as input features. The window fft size is 25 ms and hop size is 10 ms. The feature size is (25, 64, 1), following with three convolutional and pooling layers. Details can be found in the paper [26]. The kaggle leaderboard score can be 0.704 with 5 epochs, while we trained for more epochs, it can reach 0.798.

3.3. Parameter setup

The parameters of training with 1D-ConvNet and 2D-ConvNet are set as below.

For 1D-ConvNet, the loss function is a categorical cross entropy with predicted values (0∼1) and correct values (0 or 1). Adam is used as optimizer and the size of a mini-batch is set to 128. The learning rate is initially set as 1e-3. It decays when the validation accuracy does not increase for last 3 epochs with factor 0.5, while the minimum learning rate is 1e-6. Training is stopped early when validation accuracy has stopped increasing for 10 epochs. The model weights with highest validation accuracy will be saved for following predictions. For the single model, 5-fold cross validation is used to tune the parameters. 5 prediction files for test set are generated and used to do model ensemble.

For 2D-ConvNet, the loss function is same as above. Adam is used as optimizer and the size of a mini-batch is set to 16. The learning rate is initially set as 1e-3. It decays when the validation accuracy does not increase for last 4 epochs with a factor 0.5, while the minimum learning rate is 1e-6. Training is stopped early when a validation accuracy has stopped increasing for 24 epochs. The

model weights with hight validation accuracy will be stored for following predictions. For the single model, 7-fold cross validation is used to tune the parameters. 7 prediction files for test set are generated and used to do model ensemble.

3.4. Results and discussion

For 1D-ConvNet, 2 s, 3 s, 4 s, 5 s length of waveforms are extracted randomly as input. The validation accuracy (average of 5-fold CV), score and early stopping epoch numbers are listed in Table 3. As the Table shows, length of input affects little while longer waveforms as train input lead to bit better performances. For the final model ensemble, ensemble predictions with waveforms of 3 s get a higher score.

data length	val acc	LB score	stopping epoch
2 s	0.7031	0.870	58
3 s	0.7142	0.873	83
4 s	0.7205	0.869	59
5 s	0.7252	0.877	72

Table 3: Results of 1D-ConvNet with different time length input.

Different audio features are compared preliminarily with the same neural network, 2D-ConvNet. As shown in Table. 4, model trained with log-mel spectrogram and multi-resolution log-mel spectrogram get higher validation accuracy. So we use log-mel spectrogram as features for the final model ensemble. The highest public leaderboard score attained by 2D-ConvNet with log-mel spectrogram is 0.961, whose private score is 0.938.

feature	val acc
mfcc	0.7834
log-mel spectrogram	0.8662
multi-resolution log-mel spectrogram	0.8647
spectrogram	0.7878

Table 4: Results of 2D-ConvNet with different audio features.

Model ensemble is a very effective technique to increase accuracy on machine learning tasks. A good ensemble contains high performing models which are less correlated. Model ensemble methods include rank ensemble, bagging, boosting and stacking. Ranking averaging is used with predictions of 1D-ConvNet and 2D-ConvNet combined with different weighs. For our submissions, the best private leaderboard score is 0.942, while the public is 0.967, score on the whole test set is 0.947. Those scores are attained by submission 2 to challenge, details of ensemble are shown below.

- For submission 2, we took the ensemble of 5 predictions (higher validation accuracy) from 7 folds CV with 2D-ConvNet and 3 predictions from 5 folds CV with 1D-ConvNet, with weights 3:3:2:2:2:1:1:1.

With the newly released groundtruth of test set, per-class score on the whole, private and public test set can be attained as Table. 5. The accuracy for 'Scissors', 'Telephone' and 'Squeak' are too low, while overfitting is obvious for the 301 samples of public test set.

category	score(whole)	score(private)	score(public)
Oboe	0.9881	0.9853	1
Bass_drum	1	1	1
Saxophone	0.9803	0.9754	1
Chime	0.8736	0.8472	1
Electric_piano	0.9063	0.8846	1
Shatter	0.9828	0.9792	1
Bark	0.9821	0.9783	1
Acoustic_guitar	0.9667	0.9583	1
Scissors	0.7667	0.7083	1
Double_bass	1	1	1
Knock	0.9872	0.9844	1
Telephone	0.7674	0.7564	0.8148
Violin_or_fiddle	1	1	1
Gunshot_or_gunfire	0.9259	0.9379	0.8750
Burping_or_eructation	1	1	1
Clarinet	1	1	1
Computer_keyboard	0.9808	0.9762	1
Flute	0.9727	0.9659	1
Cello	0.9815	0.9773	1
Tambourine	0.9833	1	0.9167
Drawer_open_or_close	0.8793	0.8542	1
Snare_drum	1	1	1
Fart	1	1	1
Meow	0.9828	0.9792	1
Trumpet	0.9324	0.9500	0.8571
Fireworks	0.8698	0.8782	0.8333
Bus	0.8400	0.8000	1
Keys_jangling	0.9107	0.8913	1
Applause	1	1	1
Harmonica	0.9091	0.9074	0.9167
Cough	1	1	1
Gong	0.9730	0.9667	1
Glockenspiel	0.9023	0.8819	1
Tearing	0.9630	0.9545	1
Writing	0.8621	0.8542	0.9000
Squeak	0.5230	0.5069	0.6000
Microwave_oven	0.9310	0.9167	1
Laughter	0.9737	0.9677	1
Finger_snapping	1	1	1
Hi_hat	0.9829	1	0.9048
Cowbell	1	1	1

Table 5: Per-class scores on whole, private and public test set.

4. CONCLUSION

In this paper, inspired by the neural network evolutions in computer vision, we apply two deeper CNN in the DCASE 2018 task 2 - audio tagging. Though these two neural networks (1D-ConvNet and 2D-ConvNet) are not fine tuned enough till now, they showed competitive potential in this field. For 2D-ConvNet, with the same neural networks, log-mel spectrogram performs better as the input. Data augmentation like mixup, random erase are effective to overcome overfitting in this task. An easy model ensemble technique, rank averaging is used, which improved the leaderboard score slightly. More fine tuning and model ensemble techniques like stacking should be applied to get better performance, which can improve the performance and take these sound techniques to applications.

5. ACKNOWLEDGMENT

Thanks to Daisuke Niizumi, who shared lots of useful tips on Kaggle. Thanks to Zafarullah Mahmood, who gave a concise and

effective kernel as a framework of this task.

6. REFERENCES

[1] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[3] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[4] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[5] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2818–2826.

[6] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning." in *AAAI*, vol. 4, 2017, p. 12.

[7] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[8] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," *arXiv preprint arXiv:1709.01507*, vol. 7, 2017.

[9] D. Stowell, D. Giannoulis, E. Benetos, M. Lagrange, and M. D. Plumbley, "Detection and classification of acoustic scenes and events," *IEEE Transactions on Multimedia*, vol. 17, no. 10, pp. 1733–1746, 2015.

[10] G. Valenzise, L. Gerosa, M. Tagliasacchi, F. Antonacci, and A. Sarti, "Scream and gunshot detection and localization for audio-surveillance systems," in *Advanced Video and Signal Based Surveillance, 2007. AVSS 2007. IEEE Conference on*. IEEE, 2007, pp. 21–26.

[11] E. Wold, T. Blum, D. Keislar, and J. Wheaten, "Content-based classification, search, and retrieval of audio," *IEEE multimedia*, vol. 3, no. 3, pp. 27–36, 1996.

[12] H. Phan, L. Hertel, M. Maass, P. Koch, and A. Mertins, "Car-forest: Joint classification-regression decision forests for overlapping audio event detection," *arXiv preprint arXiv:1607.02306*, 2016.

[13] Q. Kong, I. Sobieraj, W. Wang, and M. Plumbley, "Deep neural network baseline for dcase challenge 2016," *Proceedings of DCASE 2016*, 2016.

[14] I. Choi, K. Kwon, S. H. Bae, and N. S. Kim, "Dnn-based sound event detection with exemplar-based approach for noise reduction," in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2016 Workshop (DCASE2016)*, 2016, pp. 16–19.

[15] Y. Xu, Q. Huang, W. Wang, P. J. Jackson, and M. D. Plumbley, "Fully dnn-based multi-label regression for audio tagging," *arXiv preprint arXiv:1606.07695*, 2016.

[16] T. Lidy and A. Schindler, "Cqt-based convolutional neural networks for audio scene classification," in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2016 Workshop (DCASE2016)*, vol. 90. DCASE2016 Challenge, 2016, pp. 1032–1048.

[17] E. Cakır, T. Heittola, and T. Virtanen, "Domestic audio tagging with convolutional neural networks," *IEEE AASP Challenge on Detection and Classification of Acoustic Scenes and Events (DCASE 2016)*, 2016.

[18] T. H. Vu and J.-C. Wang, "Acoustic scene and event recognition using recurrent neural networks," *Detection and Classification of Acoustic Scenes and Events*, vol. 2016, 2016.

[19] Y. Xu, Q. Kong, Q. Huang, W. Wang, and M. D. Plumbley, "Convolutional gated recurrent neural network incorporating spatial features for audio tagging," in *Neural Networks (IJCNN), 2017 International Joint Conference on*. IEEE, 2017, pp. 3461–3466.

[20] T. N. Sainath, R. J. Weiss, A. Senior, K. W. Wilson, and O. Vinyals, "Learning the speech front-end with raw waveform cldnns," in *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[21] W. Dai, C. Dai, S. Qu, J. Li, and S. Das, "Very deep convolutional neural networks for raw waveforms," in *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*. IEEE, 2017, pp. 421–425.

[22] S. Adavanne and T. Virtanen, "A report on sound event detection with different binaural features," *arXiv preprint arXiv:1710.02997*, 2017.

[23] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," *arXiv preprint arXiv:1710.09412*, 2017.

[24] Z. Zhong, L. Zheng, G. Kang, S. Li, and Y. Yang, "Random erasing data augmentation," *arXiv preprint arXiv:1708.04896*, 2017.

[25] J. F. Gemmeke, D. P. Ellis, D. Freedman, A. Jansen, W. Lawrence, R. C. Moore, M. Plakal, and M. Ritter, "Audio set: An ontology and human-labeled dataset for audio events," in *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*. IEEE, 2017, pp. 776–780.

[26] E. Fonseca, M. Plakal, F. Font, D. P. Ellis, X. Favory, J. Pons, and X. Serra, "General-purpose tagging of freesound audio with audioset labels: Task description, dataset, and baseline," *arXiv preprint arXiv:1807.09902*, 2018.

DCASE 2018 TASK 2: ITERATIVE TRAINING, LABEL SMOOTHING, AND BACKGROUND NOISE NORMALIZATION FOR AUDIO EVENT TAGGING

Thi Ngoc Tho Nguyen[1]*, *Ngoc Khanh Nguyen*[2], *Douglas L. Jones*[3], *Woon Seng Gan*[1],

[1] Nanyang Technological University, Electrical and Electronic Engineering Dept., Singapore, {nguyenth003, ewsgan}@ntu.edu.sg
[2] SWAT, Singapore {kylenguyen}@swatmobile.io
[3] University of Illinois at Urbana-Champaign, Dept. of Electrical and Computer Engineering, Illinois, USA, {dl-jones}@illinois.edu

ABSTRACT

This paper describes an approach from our submissions for DCASE 2018 Task 2: general-purpose audio tagging of Freesound content with AudioSet labels. To tackle the problem of diverse recording environments, we propose to use background noise normalization. To tackle the problem of noisy labels, we propose to use pseudo-label for automatic label verification and label smoothing to reduce the over-fitting. We train several convolutional neural networks with data augmentation and different input sizes for the automatic label verification process. The label verification procedure is promising to improve the quality of datasets for audio classification. Our ensemble model ranked fifth on the private leaderboard of the competition with an mAP@3 score of 0.9496.

Index Terms— Audio event tagging, Background noise normalization, Convolutional neural networks, DCASE 2018, Label smoothing, Pseudo-label

1. INTRODUCTION

The FreesoundDataset (FSD) [1] is an open general-purpose and large-scale audio dataset with the aim to promote the advancement in audio research. The data are crowd-sourced and dynamically added into the dataset via the Freesound platform, where everyone can contribute his or her own records. Such method to collect data has several advantages such that the dataset can be developed into a large dataset with a great variety of audio contents, and researchers have full access to the raw audio wave files. However, the crowd-sourcing mechanism also introduces several challenges such that unverified labels, and a wide variability in recording devices, recording environments, and audio quality.

Task 2 (audio tagging) of Dcase 2018 challenge [2] explores some of the aforementioned challenges of the FSD. In Task 2, participants are asked to classify audio clips extracted from the FSD using a subset of labels from the AudioSet Ontology [3]. There are 41 labels that cover a wide range of sound activities such as musical instruments, human sounds, domestic sounds, and animals. The training set consists of 9473 audio clips with different lengths, out of which 3710 samples have manually-verified labels and 5763 samples have non-verified labels. This is an imbalanced dataset. The number of samples for each class varies from 94 to 300 samples per class. The test set consists of 9400 audio clips, out of which around 1600 samples are used to evaluate the system performance.

Two of the main challenges of Task 2 are the label noise and the diverse nature of crowd-sourced data. A popular approach for supervised learning using cross-entropy error with noisy labels is label smoothing [4, 5]. A network is over-confident when it places all probability on a single class in the training set [5]. Label smoothing instead assigns a value less than 1 to the target class and some value to other classes in the one-hot encoded label. This technique is equivalent to regularizing the network by penalizing lower entropy output distribution [5]. The audio tagging dataset has more than 60% non-verified labels that are automatically annotated, thus it is expected that there are many samples with incorrect labels. Because the number of non-verified samples is large compared to the size of the dataset, besides label-smoothing technique, we employ pseudo-label [6, 7], which is a widely used technique in semi-supervised learning. Pseudo-labeling technique iteratively assigns pseudo-labels to some unlabeled data and use those data together with labeled data in the next training iteration. To reduce the effect of the noisy label problem in Task 2 challenge, we propose to use the pseudo-labeling technique to iteratively and automatically verify the non-verified portion of the dataset. For those samples that the classified labels from the pseudo-labelling process are different from the non-verified labels, or the classification probabilities are below a certain threshold, we employ label smoothing because we are unsure if these samples are labeled incorrectly or they are from a different varieties of the target class.

Crowd-sourced data comes from different recording environments. For example, a telephone sound can be recorded in a quiet home or on a noisy street. These background noises introduce more variability to the signal and potentially reduce the performance of the learning algorithm as the model overfits to the background noise. A common approach to mitigate the problem of background noise is multi-condition training [8, 9] where different background noises are artificially added into the signals to simulate different environments. Multi-condition training can be interpreted as a data augmentation technique, thus the performance of the model depends on how many conditions that it is trained on. To reduce the effect of background noise in crowd-sourced data, we propose to normalize the audio signal by the background noise. The background noise normalization is inspired by the psychoacoustic and physiological observations that humans and other mammals dynamically adapt to the time-varying background noise level and selectively pay attention to sound signals that are above the noise level.

*This material is based on research work supported by the IAF-ICP: Singtel Cognitive and Artificial Intelligence Lab for Enterprises@NTU under the Research Theme on Edge Intelligence.

A common pipeline for audio event classification for small audio datasets such as ESC50 [10], Urbansoudn8K [11], and TUT Acoustic Scenes [12] includes feature extraction using log-mel energy, data augmentation using pitch-shift, time-stretch, block mixing [13], random erasing/cutout [14, 15], and classification using deep neural networks [16, 17]. The data set for Task 2 is slightly larger than the Urbansound8K and has more classes. In this paper, we will follow this general pipeline, and add in the background noise normalization to build a reliable ensemble to automatically verify the training data via pseudo-labeling. Task 2 is hosted on Kaggle website. The mAP@3 scores of our best ensemble on the public leaderboard (PLB)and the final private leaderboard (PrLB) are 0.9635 and 0.9496 respectively. Our team name is "Emilia_NTU", and we ranked fifth in the competition.

We organize the paper as follows. Section II shows the building blocks of a sound event tagging model. Section III presents the automatic label verification. We report the experimental results and our DCASE 2018 submissions in Section IV. Finally, we conclude the paper in Section V.

2. BUILDING A SOUND EVENT TAGGING MODEL

We use the following parameters to convert the provided mono audio files from the time domain to the frequency domain via short time Fourier transform (STFT): sampling rate of 44.1 kHz, length-1024 FFTs, Hanning window with 50% overlap.

2.1. Preprocessing

Our preprocessing step consists of silent removal and repeating short audio clip. For each audio sample, we use librosa [18] to trim the silent part of the audio. After that, we remove frames of which the energies fall below the 10^{th} percentile of all the frame energies of that sample. Experimental results show that silent removal improves accuracy of the model. One reason might be there are many segments within an audio clips that do not contain the sound of interest. For audio samples that are shorter than the required length for deep neural networks, we repeat the signal instead of zero padding. Experimental results show that signal repeating slightly improves the model performance.

2.2. Background noise normalization

Let $\mathbf{X}$ be the audio signal in STFT domain. $\mathbf{X}$ is a matrix of size $n\_ffts/2$ x n_frames, where n_ffts is the number of FFT points and n_frames is the number of time frame. For each audio clip, the local background noise $\mathbf{x_{noise}}$ of size $n\_ffts/2$ x 1 is estimated such that $\mathbf{x_{noise}}(\mathbf{i})$ is 10^{th} percentile of signal magnitude in frequency band i. For application with streaming or continuous input, background noise can be adaptively estimated and update for different segments of audio files [19]. The signal can be normalized by the background noise as

$$\mathbf{X_{bgnorm}} = \mathbf{X}/\mathbf{x_{noise}} \quad (1)$$

Fig. 1 show an example of background noise normalization for an audio clip labeled as *Chime*. After normalization, the background noise becomes Gaussian white noise, and the signals above the background noise are highlighted. Thus the proposed normalization is promising to reduce the effect of different background noises to the classification.

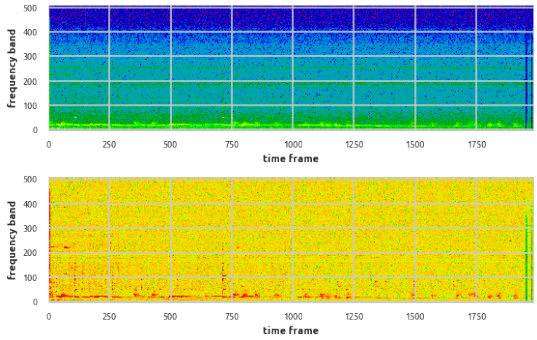


Figure 1: Example of background noise normalization for an audio clip with label *Chime*: Top is the unmodified STFT; Bottom is the normalized STFT

2.3. Feature extraction

We use librosa [18] to extract log-scale mel-spectrogram energy with the following parameters: maximum frequency of 18000 kHz and mel frequency filter bank of size 96.

2.4. Data augmentation

We use librosa [18] to generate the pitch-shift and time-stretch signal before training as the required processing time is long. The chosen pitch shift values in semitones are $[-2, -1, 1, 2]$. The chosen time-stretch ratios are [0.9, 0.95,1.05, 1.1]. We also augment data on-the-fly during training using mix-up [13], random erasing and cut-out [14, 15]. The data augmentation improves the performance of the algorithm, which is consistent with other researches [17].

2.5. Training model

We divide the provided training data into train (80%) and development set (20%). We use the competition evaluation metrics, mean average precision @ 3 (mAP@3) as the criteria to select network parameters. A convolutional neural network (CNN) that is a variant of VGG architecture [20] is used in our experiment. The CNN takes inputs as patches of log-mel spectrogram. The network architecture for input of size 128 frames and 96 mel bands is shown in Table 1. There is a total of 11 layers, which is similar to Vggish [21]. For training, we randomly extract patches from the log-mel spectrogram. The patches are normalized with the mean and standard deviation of all frames of all audio samples in the training set. For testing, we extract patches using a sliding window with hop size of 8 frames. The final prediction probability of an audio clip is the average prediction probabilities of all the extracted patches. To further improve the classification results, we extend the CNNs to take a second set of inputs which are the local mean and local standard deviation across all mel-frequency bands of each input patch. The mean and standard deviation is fed into two fully connected layers with 64 and 10 hidden neurons before being concatenated to the first fully connected layer of the CNN in Table 1.

3. AUTOMATIC LABEL VERIFICATION

The dataset contains diverse audio events with different lengths, such as short-duration gun shot and long-duration chime. To im-

Table 1: A model architecture for input of shape $(128, 96)$.

Stage	Output	Layers
Conv1	64x48	3x3, 24, BN, ReLU
		3x3, 24, BN, ReLU, maxpool, stride 2
Conv2	32x24	3x3, 48, BN, ReLU
		3x3, 48, BN, ReLU, maxpool, stride 2
Conv3	16x12	3x3, 48, BN, ReLU
		3x3, 48, BN, ReLU, maxpool, stride 2
Conv4	8x6	3x3, 64, BN, ReLU
		3x3, 64, BN, ReLU, maxpool, stride 2
Conv5	8x6	1x1, 64, BN, ReLU
fully connected	1x1	drop out, 128 fc, ReLU
		drop out, 41 fc, softmax
Number of parameters		5.48 x 10^5

prove the reliability of the label verification process, we build an ensemble of 3 CNN models. The three CNN models take patches of size $(128, 96)$, $(171, 96)$, and $(257, 96)$, which corresponds to audio segments of length 1.5 s, 2 s, and 3 s respectively. The second model use background noise normalized STFT to extract log-mel features. We combine the three models using their geometric mean which scores higher on the PLB compared to their arithmetic mean. We start the iterative training process with the verified labels. We only include the samples, of which the pseudo-labels are similar to the non-verified labels with classification probabilities greater than 0.5, to the training dataset for the next iteration. Our assumptions is that there are more samples with correct non-verified labels than those with incorrect labels. This assumption is justified since a model train on all the verified and non-verified samples produce a reasonable mAP@3 value on the PLB.

Table 2 shows the progress of the automatic label verification process using pseudo-label. The non-verified labels are considered as ground truth. We used mAP@3 as the evaluation metrics to compare the classified labels with the provided non-verified labels. The algorithm returns the best 3 classifications for each audio sample. mAP@3 returns a score of 1, $1/2$, and $1/3$, if the ground truth is matched with the best, the second best, and the third best classification respectively and returns a score of 0 otherwise. After the first iteration, our ensemble returns 3910 first best classifications, 537 second best classifications, 256 third best classifications, and 1060 incorrect classifications. Out of 3910 first best classification, 2680 samples have the classification probabilities greater than 0.5, thus they are added into the training set for the next iteration. We stop the process at iteration 4 when the number of incorrect classifications reaches a plateau. At iteration 4, we add 569 out of 576 first best classifications with the probabilities above 0.1. We relabel 152 samples out of 827 incorrect classifications, which have the classification probabilities greater than 0.5. The final training sets consists of 8039 verified samples and 1424 non-verified samples. Fig. 2 shows the histogram of the classification probabilities of the best classifications for iteration 1 and 4. At iteration 1, there are more "correct" non-verified samples, the ensemble returns high degree of confidence for many non-verified samples. At iteration 4, the distribution of the classification probabilities of the best guesses shifts toward 0. The left-over non-verified samples are either incorrectly annotated or contain different varieties of the audio class that the ensemble has not seen before. After the automatic verification process, for those 1424 left-over non-verified data, we use

Table 2: Iterative training for label verification.

# of iteration	Iter 0	Iter 1	Iter 2	Iter 3	Iter 4
# of verified samples	3710	3710	6390	7077	7470
# of new added samples	-	2680	687	393	569
# of correct pred at 1^{st} position	-	3910	1420	900	576
# of correct pred at 2^{nd} position	-	537	473	456	379
# of correct pred at 3^{rd} position	-	256	276	195	221
# of incorrect labels	-	1060	914	845	827
total # of non-verified labels	5763	5763	3083	2396	2003
mAP@3	-	0.740	0.567	0.498	0.419

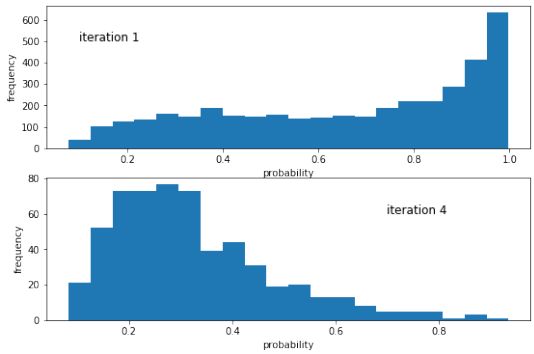


Figure 2: Histogram of classification probabilities of samples that the outputs of the pseudo-labelling ensemble are the same as the provided non-verified labels: Top is histogram for iteration 1; Bottom is histogram for iteration 4

label smoothing [22], which is defined as

$$y(i) = \begin{cases} \epsilon/k & \text{if i is none target} \\ 1 - (k-1)/k * \epsilon & \text{if i is target,} \end{cases} \quad (2)$$

in the subsequent training, where $y(i)$ is the one-hot encoded ground truth, k is the number of classes and ϵ is some small value. We sample ϵ from a uniform distribution between 0.1 and 0.3.

4. SUBMISSIONS FOR DCASE 2018 TASK 2

In this section, we discuss the results of our submissions to the Kaggle platform.

Model 1 was trained on all of the verified and non-verified data, with a segment length of 1.5 seconds, without silent removal, background normalization, data augmentation, label smoothing, and pseudo-label. Its mAP@3 values are 0.9125 and 0.8962 on the PLB and the PrLB respectively. This result shows that the non-verified data have many correct labels.

Model 2 was trained on all verified and non-verified data, with a segment length of 2 seconds, and background noise normalization. Ensemble of model 1 and 2 produced the mAP@3 values of 0.9286 and 0.9199 on the PLB and the PrLB respectively. This shows that background normalization and combining different segments lengths are helpful. In addition, the ensemble also generalizes better on the PrLB with less than 0.01 drop in the mAP@3 value. It is interesting that the mAP@3 values of model 2 on the PLB and the PrLB (0.8721 and 0.8559) are lower than those of model 1, however combining two models pushes the score significantly. From the observation that ensembles of diverse models normally perform

Table 3: Model properties.

Model ID	M1	M2	M3	M4	M5	M6	M7	M8
Input length (s)	1.5	2	1.5	1.5	2	3	1.5	2
Backgroun normalization	No	Yes	No	No	Yes	No	No	No
Silent removal	No	No	Yes	Yes	Yes	Yes	Yes	Yes
Data augmentation	No	No	Yes	Yes	Yes	Yes	Yes	Yes
Partially verified labels	No	No	No	Yes	Yes	No	No	No
Verified labels	No	No	No	No	No	Yes	Yes	Yes
Label smoothing	No	No	No	No	No	No	Yes	Yes
Additional input	No	No	No	No	No	No	No	Yes
Model mAP@3 (PLB)	0.913	0.872	0.936	0.924	0.894	0.949	0.932	0.930
Model mAP@3 (PrLB)	0.896	0.856	0.921	0.909	0.882	0.932	0.921	0.921
Ensemble mAP@3 (PLB)	0.913	0.929	0.951	-	-	0.964	0.962	0.963
Ensemble mAP@3 (PrLB)	0.896	0.920	0.936	-	-	0.947	0.948	0.950

better, we hypothesize that the background noise normalization provides some important emphasises of the signals that are not obvious in the non-normalized version.

Model 3 was trained on all verified and non-verified data, segment length of 1.5 seconds, with silent removal, data augmentation. The mAP@3 values of ensemble of model 1, 2 and 3 are 0.9507 and 0.9362 on the PLB and the PrLB respectively. This big jump on the PLB shows the importance of data augmentation and silent removal, which is consistent with the results from other researches that use small-medium size audio datasets [17, 23].

On the PrLB, our 3-model ensembles that are used for the automatically label verification process have the mAP@3 values of 0.8801, 0.9284, and 0.9222, and 0.9311 respectively at iteration 1, 2, 3 and 4. This shows that the more data we use for training, the better the validation score is. The mAP@3 value of the label-verification ensemble at iteration 3 with 7077 samples in the training set is lower than the mAP@3 value of the ensemble of model 1, 2, and 3. This implies that the learning algorithm can tolerate noisy labels and do better with more data.

The ensemble used for the label verification process consists of 3 models as mentioned in Section 3. We call these three models at iteration 4 as model 4, 5 and 6 respectively. The ensemble of model 1, 2, 3, 4, 5 and 6 has the mAP@3 values of 0.9640 and 0.9469 on the PLB. This shows that label verification using pseudo-labeling improves the mAP@3 score.

Model 7 was trained on re-labeled data after iteration 4 with a segment length of 1.5 seconds and label smoothing. Ensemble of model 1, 2, 3, 4, 5, 6, and 7 results in the mAP@3 values of 0.9623 and 0.9480 on the PLB and the PrLB respectively.

Model 8 was trained on re-labeled data after iteration 4 with a segment length of 2 seconds, label smoothing, and multiple inputs. The ensemble of model 1, 2, 3, 4, 5, 6, 7, and 8 returns the mAP@3 values of 0.9634 and 0.9496 on the PLB and the PrLB respectively which are the highest validation scores we obtained on the PLB and the PrLB. The last two ensembles do not increase the scores on the PLB but increases the score in the PrLB, which have more test data. This shows that label smoothing and multiple inputs are helpful in improving the generalization of the models.

Table 3 summaries the properties of 8 models that we used. The model mAP@3 rows report the mAP@3 value of each single model, while the ensemble mAP@3 rows report the mAP@3 value of the ensemble that includes all of the previous models. Fig. 3 shows the classification results of our 8 model ensemble. Overall, the ensemble performed reasonably well with a majority of classes have F1 scores above 0.85. There are several sound classes that achieve F1 score of 1The classes that have lowest F1 score are *Scissors*, *Chime*, *Squeak*, *Glockenspiel*, and *Firework*.

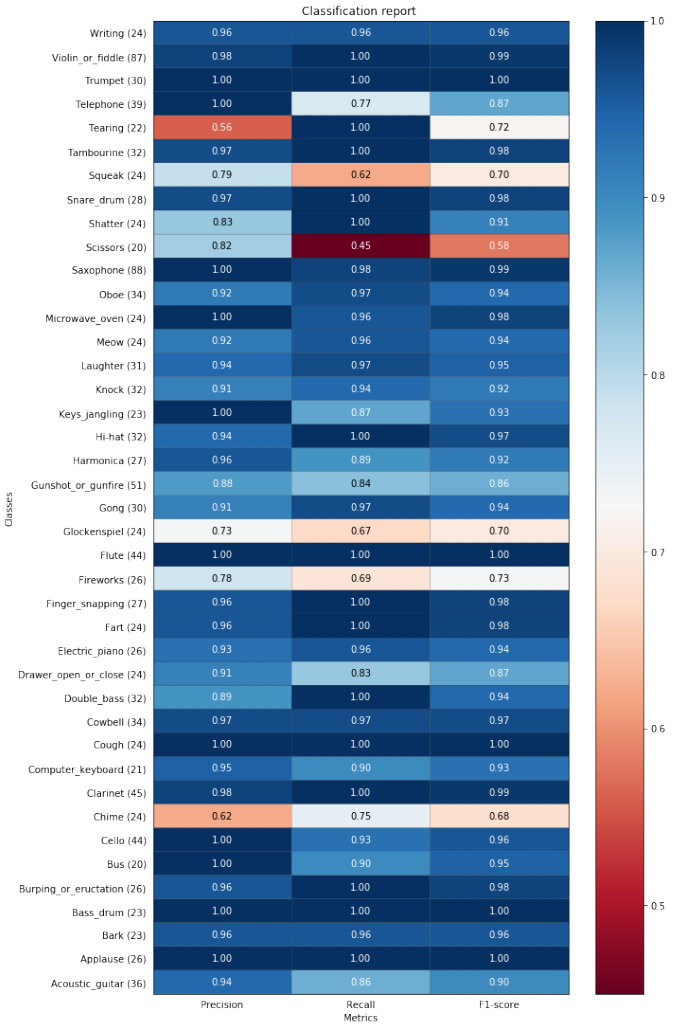


Figure 3: Classification results of an 8-model ensemble that ranked fifth in the competition

5. CONCLUSION

Datasets with high quality labels are crucial to supervised learning. However, manual annotation is expensive and time-consuming. The experimental results presented in this paper show that for small audio datasets, it is possible to increase the dataset size by training different models to automatically label new data. Manually annotation will be helpful for those "difficult" samples that the models could not resolve. Thanks to pseudo-labelling, the number of samples that need manually annotation can be reduced significantly. The number of training samples that are incorrectly annotated was unknown at the time of the competition ended, however, we can observe that the learning models are quite robust to some degree of incorrect annotation. The proposed background noise normalization introduces a useful focus of the signal to the CNNs. In addition, it is beneficial to use different input lengths for different models to improve the ensemble accuracy on datasets with diverse audio events. In conclusion, the proposed approach shows meaningful improvement compared to the baseline system.

6. ACKNOWLEDGMENT

We would like to thank Daisuke Niizumi (@daisukelab) for valuable insights on topics of audio preprocessing, data augmentation, and model training on the competition's discussion forum.

7. REFERENCES

[1] E. Fonseca, J. Pons, X. Favory, F. Font, D. Bogdanov, A. Ferraro, S. Oramas, A. Porter, and X. Serra, "Freesound datasets: a platform for the creation of open audio datasets," in *Proceedings of the 18th International Society for Music Information Retrieval Conference (ISMIR 2017)*, Suzhou, China, 2017, pp. 486–493.

[2] E. Fonseca, M. Plakal, F. Font, D. P. W. Ellis, X. Favory, J. Pons, and X. Serra, "General-purpose tagging of freesound audio with audioset labels: Task description, dataset, and baseline," 2018, submitted to DCASE2018 Workshop. [Online]. Available: https://arxiv.org/abs/1807.09902

[3] J. F. Gemmeke, D. P. W. Ellis, D. Freedman, A. Jansen, W. Lawrence, R. C. Moore, M. Plakal, and M. Ritter, "Audio set: An ontology and human-labeled dataset for audio events," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, March 2017, pp. 776–780.

[4] G. Pereyra, G. Tucker, J. Chorowski, Ł. Kaiser, and G. Hinton, "Regularizing neural networks by penalizing confident output distributions," *arXiv preprint arXiv:1701.06548*, 2017.

[5] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2818–2826.

[6] D.-H. Lee, "Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks," in *Workshop on Challenges in Representation Learning, ICML*, vol. 3, 2013, p. 2.

[7] H. Wu and S. Prasad, "Semi-supervised deep learning using pseudo labels for hyperspectral image classification," *IEEE Transactions on Image Processing*, vol. 27, no. 3, pp. 1259–1270, 2018.

[8] D. Garcia-Romero, X. Zhou, and C. Y. Espy-Wilson, "Multi-condition training of gaussian plda models in i-vector space for noise and reverberation robust speaker recognition," in *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*. IEEE, 2012, pp. 4257–4260.

[9] M. L. Seltzer, D. Yu, and Y. Wang, "An investigation of deep neural networks for noise robust speech recognition," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, May 2013, pp. 7398–7402.

[10] K. J. Piczak, "Esc: Dataset for environmental sound classification," in *Proceedings of the 23rd ACM international conference on Multimedia*. ACM, 2015, pp. 1015–1018.

[11] J. Salamon, C. Jacoby, and J. P. Bello, "A dataset and taxonomy for urban sound research," in *Proceedings of the 22Nd ACM International Conference on Multimedia*, ser. MM '14. New York, NY, USA: ACM, 2014, pp. 1041–1044. [Online]. Available: http://doi.acm.org/10.1145/2647868.2655045

[12] A. Mesaros, T. Heittola, and T. Virtanen, "Tut database for acoustic scene classification and sound event detection," in *Signal Processing Conference (EUSIPCO), 2016 24th European*. IEEE, 2016, pp. 1128–1132.

[13] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," *arXiv preprint arXiv:1710.09412*, 2017.

[14] T. DeVries and G. W. Taylor, "Improved regularization of convolutional neural networks with cutout," *arXiv preprint arXiv:1708.04552*, 2017.

[15] Z. Zhong, L. Zheng, G. Kang, S. Li, and Y. Yang, "Random erasing data augmentation," *arXiv preprint arXiv:1708.04896*, 2017.

[16] K. J. Piczak, "Environmental sound classification with convolutional neural networks," in *Machine Learning for Signal Processing (MLSP), 2015 IEEE 25th International Workshop on*. IEEE, 2015, pp. 1–6.

[17] J. Salamon and J. P. Bello, "Deep convolutional neural networks and data augmentation for environmental sound classification," *IEEE Signal Processing Letters*, vol. 24, no. 3, pp. 279–283, March 2017.

[18] B. M. et. al., "librosa/librosa: 0.6.1," May 2018. [Online]. Available: https://doi.org/10.5281/zenodo.1252297

[19] R. Martin, "Noise power spectral density estimation based on optimal smoothing and minimum statistics," *IEEE Transactions on speech and audio processing*, vol. 9, no. 5, pp. 504–512, 2001.

[20] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[21] S. Hershey, S. Chaudhuri, D. P. W. Ellis, J. F. Gemmeke, A. Jansen, C. Moore, M. Plakal, D. Platt, R. A. Saurous, B. Seybold, M. Slaney, R. Weiss, and K. Wilson, "Cnn architectures for large-scale audio classification," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017. [Online]. Available: https://arxiv.org/abs/1609.09430

[22] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning*. MIT press Cambridge, 2016, vol. 1.

[23] S. Mun, S. Park, D. K. Han, and H. Ko, "Generative adversarial network based acoustic scene training set augmentation and selection using SVM hyper-plane," in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2017 Workshop (DCASE2017)*, November 2017, pp. 93–102.

ACOUSTIC EVENT SEARCH WITH AN ONOMATOPOEIC QUERY: MEASURING DISTANCE BETWEEN ONOMATOPOEIC WORDS AND SOUNDS

Shota Ikawa[1], *Kunio Kashino*[1,2]

[1] Graduate School of Information Science and Technology, University of Tokyo, Japan
[2] NTT Communication Science Laboratories, NTT Corporation, Japan

ABSTRACT

As a means of searching for desired audio signals stored in a database, we consider using a string of an onomatopoeic word, namely a word that imitates a sound, as a query, which allows the user to specify the desired sound by verbally mimicking the sound or typing the sound word, or the word containing sounds similar to the desired sound. However, it is generally difficult to realize such a system based on text similarities between the onomatopoeic query and the onomatopoeic tags associated with each section of the audio signals in the database. In this paper, we propose a novel audio signal search method that uses a latent variable space obtained through a learning process. By employing an encoder-decoder onomatopoeia generation model and an encoder model for the onomatopoeias, both audio signals and onomatopoeias are mapped within the space, allowing us to directly measure the distance between them. Subjective tests show that the search results obtained with the proposed method correspond to the onomatopoeic queries reasonably well, and the method has a generalization capability when searching. We also confirm that users preferred the audio signals obtained with this approach to those obtained with a text-based similarity search.

Index Terms— audio signal search, onomatopoeia, latent variable, encoder-decoder model

1. INTRODUCTION

Recently, a large amount of audio data is being accumulated in local storage or on the Internet, and the demand for an audio signal search technique has been increasing. Audio signal search methods can be divided into two types according to query types: search with an audio query and search with a text query.

For the former, searches based on audio feature matching is widely utilized [1]. However, except for the cases such as audio fingerprinting, there are generally many cases where the audio signal or feature is difficult to obtain to use as a query. For example, sound engineers who want to find specific sound effects in a sound database will not have the desired signal that can be used as a query. For the latter type of search, sound classification or description tags must be attached to acoustic signals in advance. For example, a video hosting service can use metadata, the anchor texts of incoming links, and comments as text information. However, it is widely known that automatic audio classification or description is not a simple task [2], and therefore, this approach sometimes requires a lot of human labor.

Against this background, we propose the use of onomatopoeias as audio search queries. The application we have in mind is a generic sound search system that allows users to find or locate their desired sounds. For example, it would be useful to be able to spot specific audio samples or events, such as birds' songs, machine failure sounds, or accident sounds, from among a vast amount of stored data.

Onomatopoeias are the words that imitate non-speech sounds within the pronunciation of a certain language system, and there are two modes: written and spoken. Onomatopoeias are widely seen in many languages, including English, Chinese, and Japanese, and they effectively support our daily communication. The use of onomatopoeias helps us to express acoustic information in a form that others can easily understand [3]. In previous studies, they have been effectively used for intuitive audio searches [4], and as a kind of classification tags for acoustic events [5, 6]. Up to now, most research using onomatopoeias for audio search has been text-based, which means it was based on the textual similarities between the onomatopoeic query and the onomatopoeic tags attached to the acoustic signals in advance [7]. However, as detailed in the following section, this approach poses several problems.

To solve these problems, here we propose a method that takes advantage of a latent space. The space is obtained through the learning process of an encoder-decoder model [8, 9] for onomatopoeia generation [10]. The space can be sufficient to allow it to be shared by onomatopoeic and audio signal encoders. This allows us to directly measure the distance between a written or spoken onomatopoeia and a section of an audio signal, which means that we can perform a similarity search for audio signals with an onomatopoeia query, without audio classification, description or transcription.

The rest of this paper is organized as follows. Section 2 discusses the problems of the existing text-based audio search methods. Section 3 introduces our method. Section 4 evaluates our proposed system. Section 5 concludes the paper.

2. PROBLEMS WITH TEXT-BASED AUDIO SEARCH

Previous work on audio signal search with an onomatopoeic query has usually been based on the similarity between the query text and the onomatopoeia tags associated with each audio signal in the database. In addition to the preprocessing, or human labor, needed to attach such tags in the database, this approach essentially poses the following problems.

First, many search results can give the exactly same similarity to a query. This is due to the fact that onomatopoeias are highly-compressed, coarsely-quantized representation of sounds. This makes it difficult to obtain an appropriately ordered result list. As the database grows in size, the usability can be seriously degraded.

Second, it is generally difficult to determine one correct onomatopoeic tag for an audio signal; that is, one audio signal can be described as different onomatopoeias, depending on the listeners.

This is due to the intrinsic ambiguity in onomatopoeic expressions [11]. For this reason, the quality and quantity of the onomatopoeic tags in the database greatly affect the accuracy, efficiency and usability of the search.

3. SEARCH BASED ON LATENT VARIABLES

3.1. Audio search problem definition

Let $\boldsymbol{z}_{\boldsymbol{x}}$ be a latent variable derived from an audio signal $\boldsymbol{x}$, and $\boldsymbol{z}_{\boldsymbol{l}}$ be an onomatopoeic latent variable derived from an onomatopoeia. Here, a latent variable is a fixed-dimensional vector. When $\boldsymbol{z}_{\boldsymbol{x}}, \boldsymbol{z}_{\boldsymbol{l}}$ are two points in the shared latent space $V \subset \mathbb{R}^n$, the distance between the audio signal and the onomatopoeia is defined as follows:

$$D(\boldsymbol{x}, \boldsymbol{l}) \equiv \|\boldsymbol{z}_{\boldsymbol{x}} - \boldsymbol{z}_{\boldsymbol{l}}\|. \tag{1}$$

$\|\cdot\|$ is norm on V. Here, audio search is defined as finding the nearest audio signals to a query based on the distance given in Eq. 1. Hereafter, we assume the query is given in the form of a written onomatopoeia, although the same framework can be applied to the case of spoken onomatopoeias.

3.2. Extracting latent variables

We employ an onomatopoeia generation model to calculate $\boldsymbol{z}_{\boldsymbol{x}}$ from the corresponding audio signal. The model is based on the idea that an onomatopoeia phoneme string $\boldsymbol{l}$ is generated according to a conditional probability distribution $p(\boldsymbol{l}|\boldsymbol{z}_{\boldsymbol{x}})$. That is, it generates the onomatopoeia string $\bar{\boldsymbol{l}}$, which has the highest probability given an audio signal.

$$\bar{\boldsymbol{l}} = \arg\max_{\boldsymbol{l}} p(\boldsymbol{l}|\boldsymbol{z}_{\boldsymbol{x}}) \tag{2}$$

This estimation is decomposed into: (1) the estimation of a mapping $f : \boldsymbol{x} \to \boldsymbol{z}_{\boldsymbol{x}}$, namely the extraction of a latent variable from an audio signal x, and (2) the generation of the most plausible onomatopoeia $\bar{\boldsymbol{l}}$ given the latent variable $\boldsymbol{z}_{\boldsymbol{x}}$. The former step is used to obtain $\boldsymbol{z}_{\boldsymbol{x}}$ from x.

The onomatopoeic latent variable $\boldsymbol{z}_{\boldsymbol{l}}$ is extracted from an onomatopoeia l as follows. With the onomatopoeia generation model, the probability of $\boldsymbol{z}_{\boldsymbol{l}}$ is given by the conditional probability density distribution $p(\boldsymbol{z}|\boldsymbol{l})$, which is the likelihood function of $\boldsymbol{l}$. Thus, we regard the conditional expectation of $\boldsymbol{z}$ as the onomatopoeic latent variable $\boldsymbol{z}_{\boldsymbol{l}}$. That is, a mapping $g : \boldsymbol{l} \to \boldsymbol{z}_{\boldsymbol{l}}$, namely the extraction of the latent variable from an onomatopoeia, is formulated as:

$$g(\boldsymbol{l}) = \int_V \boldsymbol{z} p(\boldsymbol{z}|\boldsymbol{l}) d\boldsymbol{z}. \tag{3}$$

3.3. Solution with neural networks

Ikawa *et al.* [10] proposed using an encoder-decoder model to obtain onomatopoeic representation from sounds. The encoder corresponds to the mapping f and the decoder corresponds to the estimation of $\bar{\boldsymbol{l}}$ from $\boldsymbol{z}_{\boldsymbol{x}}$, and they are estimated simultaneously.

We used the encoder-decoder model shown in Figure 1. The audio latent variable $\boldsymbol{z}_{\boldsymbol{x}}$ is calculated from acoustic features. Hereafter, we call this part an audio signal encoder. Then, the initial states of the decoder-LSTMs are calculated from $\boldsymbol{z}_{\boldsymbol{x}}$. Here, the dimension of the latent space V is determined by the number of units of the corresponding layer of the neural network. Using tanh as the activation function, each element of $\boldsymbol{z}$ takes the value $[-1, 1]$.

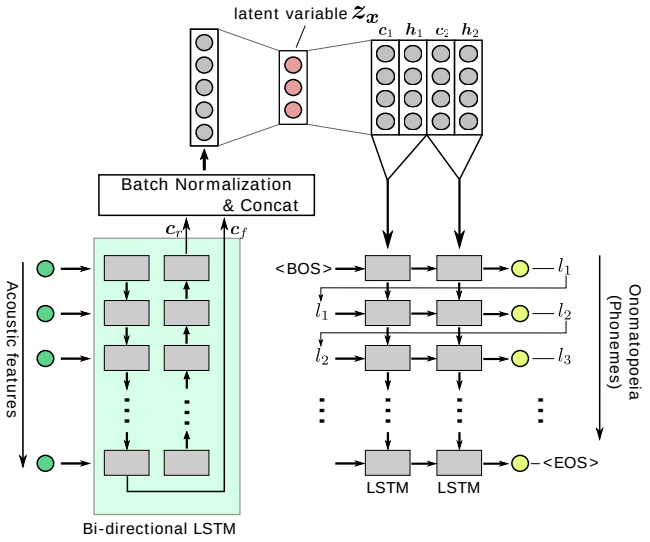


Figure 1: Block diagram of the audio signal encoder-decoder model.

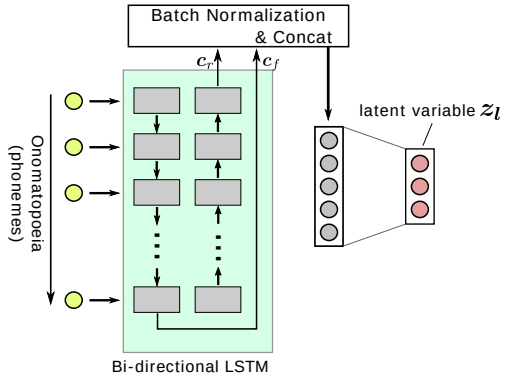


Figure 2: Block diagram of onomatopoeic encoder model.

The mapping g can also be obtained using a neural network (hereafter, onomatopoeic encoder). Figure 2 shows the structure of the onomatopoeic encoder. The estimated mapping $\hat{g} = g_{\hat{\boldsymbol{\theta}}}$ is acquired based on the learned parameters of the onomatopoeic encoder $\hat{\boldsymbol{\theta}}$. With the estimated audio signal encoding mapping $\hat{f}$, the loss function used to train the onomatopoeic encoder is written as:

$$\mathcal{L}(\boldsymbol{\theta}) = \|g_{\boldsymbol{\theta}}(\boldsymbol{l}) - \hat{f}(\boldsymbol{x})\|, \tag{4}$$

where the definition of norm is the same as in Eq. (1).

3.4. Audio signal search

Using the estimated mappings $\hat{f}, \hat{g}$, the audio signal search is realized by measuring the distances between the onomatopoeic query and each audio signal in the database:

$$\hat{D}(\boldsymbol{x}, \boldsymbol{l}) = \|\hat{f}(\boldsymbol{x}) - \hat{g}(\boldsymbol{l})\|. \tag{5}$$

The neural networks are trained with a set of audio signals associated with onomatopoeic tags. Unlike the existing text-based

Table 1: Experimental conditions

LSTM cells	512
Batchsize	256
Output phoneme labels	32
Optimizer	ADAM [14]
MFCC dimension	20
FFT window (MFCC)	2048 samples
FFT shift (MFCC)	512 samples

search methods described in section 2, once the test set is given, our method does not need an onomatopoeic tag for any of the audio signals in the database.

4. EXPERIMENTS

We evaluate the proposed method from two standpoints: the appropriateness of the search results and by comparing it with a text-based search. In both cases, the task is to find the nearest neighbor signal. For an audio signal database $X = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\}$ and an onomatopoeic query $\boldsymbol{l}$, the nearest neighbor signal $\bar{\boldsymbol{x}}(\boldsymbol{l})$ is represented as:

$$\bar{\boldsymbol{x}}(\boldsymbol{l}) = \underset{\boldsymbol{x}_i \in X}{\arg\min}\, D(\boldsymbol{x}_i, \boldsymbol{l}). \tag{6}$$

4.1. Dataset

We used a subset of the audio signals contained in the Real World Computing Partnership (RWCP) sound scene database [12] to train the neural networks. The database includes various sound samples recorded without background noise in an unechoic environment and digitized at 48 kHz, with linear PCM of 16 bit accuracy.

For the training, we chose 709 signals, including those made by bells, coins, and hitting wood with a stick. The number of the class labels (bell, coin, etc.) was 81, and 7 to 10 signals were sampled for each class.

To build the training set, onomatopoeic tags were collected from human listeners. Considering the ambiguity of onomatopoeia, multiple onomatopoeic tags were attached to each audio signal. To accomplish this, 73 Japanese speakers were asked to produce three onomatopoeias for each sound using katakana, which is a Japanese syllabary. Each katakana answer was converted to a string based on the International Phonetic Alphabet (IPA) [13] and used as an onomatopoeic tag. In Japanese, onomatopoeias are usually written in katakana, and it is straightforward to convert katakana to IPA, and vice versa. We associated 12 onomatopoeias for each audio signal in the dataset.

Note that we used the IPA symbols as a simple universal representation of pronunciation in the experiments, but any phonogram sequences, or texts, can be used in our framework.

4.2. Learning of the encoder-decoder onomatopoeia generation model

Table 1 lists the experimental conditions. For simplicity, we used a series of mel-frequency cepstral coefficients (MFCC) as the input. As output phonemes, we used 29 kinds of symbols that consist of the standard IPA phonetic symbols and Japanese-specific ones: "ɴ" for moraic nasal, "H" for the second mora of a long vowel and "Q" for a moraic silence when emphatic. In addition, we used three special symbols: "BOS (beginning of the sequence)," "EOS (end of the sequence)," and "UNK (unknown)."

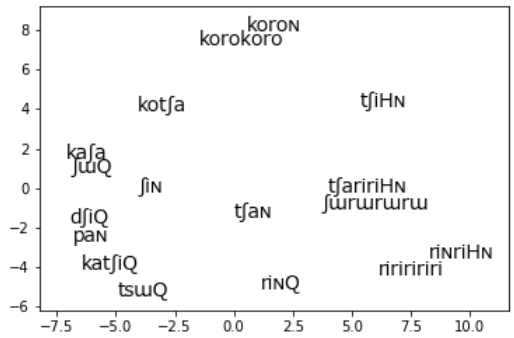


Figure 3: Onomatopoeic latent variables whose dimensions are reduced from 128 to 2 by using PCA. It is shown that onomatopoeias with similar characteristics (e.g. "koroɴ" and "korokoro") are closely located.

From the preliminary experiments, we chose 128 as the number of latent variable dimensions. After 34 epochs of learning, the audio signal encoder-decoder model achieved a 9.9% word error rate and a 4.0% mean phoneme error rate for an onomatopoeia generation task [10], with a test set consisting of 101 audio signals, which were again sampled from the RWCP dataset excluding the ones used for the network training.

4.3. Learning of the onomatopoeic encoder

The same training data were used for training the onomatopoeic encoder as in the previous section. L1-norm was employed as the loss function (Eq. (4)). Figure 3 shows an example of the resulting distribution of onomatopoeic latent variables. It is observed that the onomatopoeias with similar characteristics are localized to each other in the latent space.

4.4. Experimental setups and results

Experiment 1: Suitability of signals found for queries

This experiment was designed to confirm whether the found signals correctly corresponded to the onomatopoeic queries.

The subjects were presented with an onomatopoeia in katakana on a display, which was a query, and then with an audio signal, which was a result of the nearest neighbor search using the proposed method. They were then asked to choose one of five options: "very suitable," "relatively suitable," "neutral," "relatively unsuitable," and "very unsuitable." We performed the experiment using two different audio databases:

- A database consisting of the above-mentioned "training" set sampled from the RWCP database (hereafter, the RWCP test set)
- A database consisting of the sounds sampled from another dataset (hereafter, the external test set)

The former was used to verify the basic behavior, and the latter was used to evaluate the generalization performance of the proposed method.

For the external test set, we used part of Free Sound Dataset Kaggle 2018 [15], which is a subset of FSD [16] and is used for

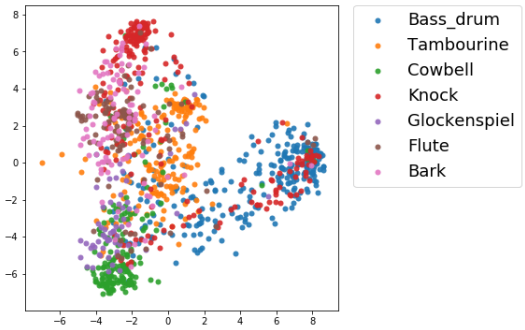


Figure 4: Distributions of audio latent variables for the external test set. Dimensions of each latent variable are reduced from 128 to 2 by using PCA. It is observed that the samples that belong to the same class tend to be localized in this space.

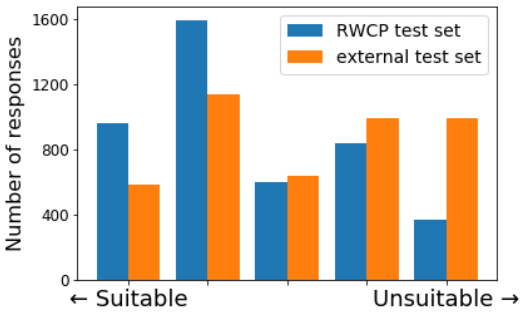


Figure 5: Suitability of the audio signals presented by the search

the general-purpose audio tagging task in the DCASE 2018 CHALLENGE. There were 11,719 audio signals. The MFCC sequence for each audio signal was calculated according to Table 1 after converting the sampling frequency from 44.1 kHz to 48 kHz by using FFmpeg [17]. Figure 4 shows the distributions of the audio latent variables of some of the external test set obtained by the trained model visualized by PCA.

We chose 217 Japanese onomatopoeias as the queries for each test set. There were 20 subjects, and the total number of responses for each test set was 4,340.

Figure 5 shows the result. The most frequent response for both test sets was "relatively suitable". For the RWCP test set, 58.7% of the responses were "suitable," showing that the proposed method worked effectively. For the external test set, the "suitable" responses amounted to 39.7%, which is fewer than the RWCP case. This is because the number and variations of audio signals included in the external test set is much greater than that of the training set. However, this still means that the proposed model has a generalization ability even for the external test set, because if the audio samples were randomly presented in this test, most responses must have been "very (or relatively) unsuitable".

Experiment 2: Comparison of the proposed method and the text-based method

We used the audio database that consisted of the same audio samples as those used in the network training in order to evaluate whether the search results obtained with the proposed method were

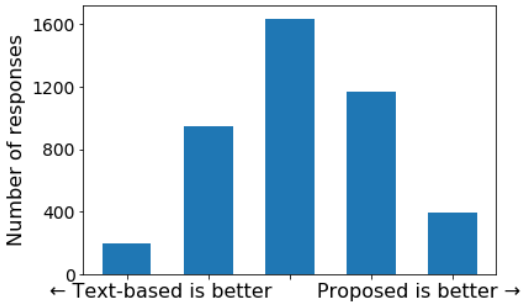


Figure 6: Comparison of the proposed method and the text-based method

preferable to those obtained with the text-based method. The subjects were presented with one onomatopoeia on a display and two audio signals, "A" and "B." They were then asked to choose one of five options: "A is much better," "A is relatively better," "no difference," "B is relatively better," and "B is much better." Either "A" or "B" (randomly selected) was the search result obtained with the proposed method and the other was the one obtained with the text-based method.

The text-based method in this experiment was based on the similarity measured by the edit (Levenshtein) distance between the IPA strings. In the audio database, multiple audio signals can correspond to the same onomatopoeic tag, yielding multiple search results for one query with the same similarity. In such cases, one signal was randomly chosen as the result.

As in Experiment 1, 217 onomatopoeias were used as the queries, and 20 subjects undertook the evaluation. The total number of responses was 4,340.

Figure 6 shows the result. It is shown that the proposed method is preferable to the text-based method, since the distribution is clearly biased to the right from the center. For a quantitative analysis, we assign scores of 2, 1, 0, -1, -2, according to the five kinds of responses, so that the score become larger when the proposed method receives a higher evaluation. The mean of the score appears to be 0.145, and from the t test, it is not 0 at the 1% significance level. This means that the proposed method produced significantly better results than the text-based method.

5. CONCLUSION

This paper proposed a novel method for finding audio signals with an onomatopoeic query. With our method, the distance between an audio signal and an onomatopoeic symbol sequence is directly measured in the latent space. We showed the effectiveness of the proposed method by performing subjective experiments. This paper focused on the use of written onomatopoeias, but we expect that it is straightforward to train the network so that it accepts spoken onomatopoeias as queries. Our future work will also include tests with languages other than Japanese, and a usability study with practical senarios.

6. ACKNOWLEDGMENTS

We thank Prof. Hiroshi Saruwatari and Dr. Shinnosuke Takamichi for their valuable comments and support.

7. REFERENCES

[1] E. Wold, T. Blum, D. Keislar, and J. Wheaten, "Content-based classification, search, and retrieval of audio," *IEEE MultiMedia*, vol. 3, no. 3, pp. 27–36, Fall 1996.

[2] J. F. Gemmeke, D. P. W. Ellis, D. Freedman, A. Jansen, W. Lawrence, R. C. Moore, M. Plakal, and M. Ritter, "Audio set: An ontology and human-labeled dataset for audio events," in *Proc. IEEE ICASSP 2017*, New Orleans, LA, 2017.

[3] F. Wang, H. Nagano, K. Kashino, and T. Igarashi, "Visualizing video sounds with sound word animation to enrich user experience," *IEEE Transactions on Multimedia*, vol. 19, no. 2, pp. 418 – 429, 2017.

[4] T. A. Sanae Wake, "Sound retrieval with intuitive verbal descriptions," *IEICE Trans. Inf. and Syst.*, vol. E84-D, no. 11, pp. 1568 – 1576, 2001.

[5] S. Sundaram and S. S. Narayanan, "Classification of sound clips by two schemes: using onomatopoeia and semantic labels," in *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME)*, Hannover, Germany, jun 2008, pp. 1341–1344.

[6] K. Hayashida, Y. Mizoguchi, J. Ogawa, M. Morise, T. Nishiura, and Y. Yamashita, "The acoustic sound field dictation with hidden markov model based on an onomatopeia," vol. 5, 01 2010.

[7] K. Okamoto, R. Yamanishi, and M. Matsushita, "Sound-effects exploratory retrieval system based on various aspects," *IEEJ Transactions on Electronics, Information and Systems*, vol. 136, no. 12, pp. 1712–1720, 2016.

[8] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2014, pp. 3104–3112. [Online]. Available: http://papers.nips.cc/paper/5346-sequence-to-sequence-learning-with-neural-networks.pdf

[9] K. Cho, B. van Merrienboer, Ç. Gülçehre, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," *CoRR*, vol. abs/1406.1078, 2014. [Online]. Available: http://arxiv.org/abs/1406.1078

[10] S. Ikawa and K. Kashino, "Generating sound words from audio signals of acoustic events with sequence-to-sequence model," in *Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 346 – 350.

[11] K. Ishiguro, Y. Tsubota, and H. Okuno, "Automatic transformation of environmental sounds into sound-imitation words based on japanese syllable structure," in *Proc. EUROSPEECH*, 2003, pp. 3185–3188.

[12] S. Nakamura, K. Hiyane, F. Asano, T. Yamada, and T. Endo, "Data collection in real acoustical environments for sound scene understanding and hands-free speech recognition," in *Proc. EUROSPEECH*, Sep. 1999, pp. 2255–2258.

[13] I. P. Association, *Handbook of the International Phonetic Association: A guide to the use of the International Phonetic Alphabet*. Cambridge University Press, 1999.

[14] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proceedings of the 3rd International Conference on Learning Representation (ICLR)*, 2015.

[15] https://www.kaggle.com/c/freesound-audio-tagging/data.

[16] E. Fonseca, J. Pons, X. Favory, F. Font, D. Bogdanov, A. Ferraro, S. Oramas, A. Porter, and X. Serra, "Freesound datasets: a platform for the creation of open audio datasets," in *Proceedings of the 18th International Society for Music Information Retrieval Conference (ISMIR 2017)*, Suzhou, China, 2017, pp. 486–493.

[17] https://www.ffmpeg.org/.

SOUND EVENT DETECTION FROM WEAK ANNOTATIONS: WEIGHTED GRU VERSUS MULTI-INSTANCE LEARNING

Léo Cances, Thomas Pellegrini, Patrice Guyot

IRIT, Université de Toulouse, CNRS, Toulouse, France
{leo.cances,thomas.pellegrini,patrice.guyot}@irit.fr

ABSTRACT

In this paper, we address the detection of audio events in domestic environments in the case where a weakly annotated dataset is available for training. The weak annotations provide tags from audio events but do not provide temporal boundaries. We report experiments in the framework of the task four of the DCASE 2018 challenge. The objective is twofold: detect audio events (multi-categorical classification at recording level), localize the events precisely within the recordings. We explored two approaches: 1) a "weighted-GRU" (WGRU), in which we train a Convolutional Recurrent Neural Network (CRNN) for classification and then exploit its frame-based predictions at the output of the time-distributed dense layer to perform localization. We propose to lower the influence of the hidden states to avoid predicting a same score throughout a recording. 2) An approach inspired by Multi-Instance Learning (MIL), in which we train a CRNN to give predictions at frame-level, using a custom loss function based on the weak label and statistics of the frame-based predictions. Both approaches outperform the baseline of 14.06% in F-measure by a large margin, with values of respectively 16.77% and 24.58% for combined WGRUs and MIL, on a test set comprised of 288 recordings.

Index Terms— Sound event detection, weakly supervised learning, multi-instance learning, convolutional neural networks, weighted gate recurrent unit

1. INTRODUCTION

The coming of Deep Learning [1] has opened a new era in the domain of artificial intelligence. Deep neural networks in particular became the state of the art in many application domains involving classification and detection tasks. Most often, these improvements rely on the availability of ever-growing annotated datasets to train the models. While many previous works that heavily rely on supervised training based on a precise manual annotation, new challenges arise from the use of large datasets without supervision.

Recently, different databases of Terabytes of data have been released by Google. The Audioset database provides a large set of audio data extracts from video [2]. Annotations for audio events are mainly based on tags by YouTube users and do not contain temporal information.

In that scope, the DCASE challenge includes a task on sound event detection in domestic environment [3]. This task proposes a framework to build a system that aims at detecting audio events from a set of 10 classes of sound events forming a subset of Audioset. More precisely, the aim is to provide starting and ending boundaries of the audio events (strong labels) while the training set relies only on global tags (weak labels). As mentioned in [3], the duration of the targeted sounds depends heavily on their class. For example, the class *vacuum cleaner* contains mostly audio events of 10 seconds, while the class *dog* is mainly composed of sounds shorter than half-second.

Sound event detection (SED) has been deeply investigated [4]. In real life, sound events overlap to produce a mixture. In the same way, the current challenge aims at detecting overlapping sound events, referred as polyphonic SED. Polyphonic SED covers a wide set of applications including ecology [5] and surveillance [6]. The detection of domestic sounds provides interesting clues for health applications [7] and Intelligent Virtual Assistants such as Google Home or Amazon Echo.

Ongoing research works on SED are mostly based on a Deep Neural Networks (DNNs). They include fully-connected DNNs [8], Convolutional Neural Networks (CNNs) [9] and Recurrent Neural Networks (RNNs) [10]. Most of recent approaches are based on a combination of layers including these different elements [11]. In particular the issue of audio event detection using weakly labeled data was addressed in [12, 13] and formulated into a Multi-Instance Learning (MIL) problem.

The baseline method provided by the challenge organizers relied on two Convolutional Recurrent Neural Networks (CRNNs): the first one for file-level audio classification (weak labels), the second one for the localization of the previously detected events within the recordings (strong labels). We explored two separate approaches that both outperformed the baseline.

Firstly, we modify the recurrent layer of a CRNN to be able to weight the influence of the hidden state of the recurrent cells. Secondly, we generalize to our multiclass classification problem a new loss function inspired by Multi-Instance Learning (MIL), very recently proposed for singing bird localization in [14].

Section 2 describes the first approach, that we will refer to as "weighted Gated Recurrent Unit" (WGRU), followed by a section describing the second approach (MIL). We report the experimental setup in Section 4, and finally analyze the results and limitations of the two approaches.

2. WEIGHTED GATED RECURRENT UNIT (WGRU)

2.1. Temporal detection from an adapted baseline method

As mentioned above, the baseline system is based on two convolutional recurrent neural networks (CRNN). The first CRNN detects the presence/absence of the ten sound events of interest at file-level. Then, a second CRNN is used for localization. Still, we may assume that the temporal information required for localization is reachable from the first CRNN. In this way, after the classification training of the first CRNN on weak labels, we propose to simply remove its final global average pooling layer in order to get frame-based predictions used for detection. This modified model produces frame-

level predictions thank to its time-distributed dense comprised of ten sigmoid units corresponding to the ten classes of interest.

In the following, we will focus on the recurrent layer of this CRNN, since modifications of its internal functioning had to be made to make localization possible.

2.2. Recurrent Neural Networks

Recurrent Neural Networks capture temporal behavior in sequential data [15]. The hidden state of a RNN cell depends on x_t, the incoming output of the previous layer at time t, and h_{t-1}, its hidden state at time $t-1$, as defined in Equation 1:

$$h_t = g(Wx_t + Uh_{t-1} + b) \tag{1}$$

where g is a point-wise activation function (hyperbolic tangent function in our case); and W and U are weight matrices to be learned together with the bias b.

2.3. Weighted RNN

RNNs have proven to model sound events efficiently, since these often have an underlying sequential structure [10].However, some audio event classes have different typical durations, as described in [3]. Long-duration sounds (*vacuum cleaner*, *running water*) are expected to be easier to model by an RNN than short sounds (*dog*, *cat)*. In order to adjust our models to different kinds of sounds, we propose a new adaptation of RNNs. This approach aims at configuring different kinds of temporal behavior according to the class of a sound event. As a first attempt in that direction, we weight the influence of the hidden state of the recurrent cells by a factor $\omega \leq 1$, which we set globally for the classes on a development subset. This modifies the impact of sequentiality in the RNN, as formulated in equation 2. Figure 1 shows the impact of the weight on a cell at time t, colored in blue. The rational behind lowering the impact of the hidden states lies in the fact that the CRNN is trained to detect a sound event at file level. Thus, if an event is detected at the beginning of the file, the hidden states are expected to keep that information throughout the file even if the detected event is not present in the whole file, and the localization afterwards will fail.

The baseline method is based on Gated Recurrent Units (GRU) [16]. In the following, we will use weighted RNN models using a GRU layer that we will be referred to as Weighted GRU (WGRU).

$$h_t = g(Wx_t + U\boldsymbol{\omega}h_{t-1} + b) \tag{2}$$

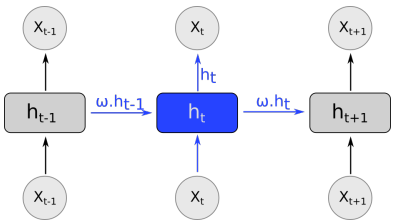


Figure 1: Configuration of the recurrent links between RNN cells according to the weight ω.

The next question that arises is how to set ω. We decided to set a single value for all the sound types based on the localization performance measured on a held-out validation subset. A weight of $\omega = 1$ corresponds to a standard GRU. A lower weight is expected to be more adapted for events of short duration.

The use of different weights does not require to retrain a model. We simply replace the GRU layer by a WGRU at inference time. Thus, we perform two forward passes with a single model: one with GRU for classification, one with WGRU for localization. Finally, we are able to produce temporal predictions with different weights, and eventually combine them to improve the localization.

3. MULTI-INSTANCE LEARNING

Another approach is related to the Multi-Instance Learning (MIL) paradigm [17], which handles cases with weak labels. In our case, we need to make predictions at frame-level, whereas the reference tags are at file-level. When a recording is labeled with the *Cat* class, for instance, not all the acoustic frames are positive with respect to that class. Thus, we are in presence of both positive and negative instances at frame-level. A straightforward but suboptimal solution is the so-called "false strong labeling", in which we consider that all the frames are positive for a given class. MIL consists instead of considering that at least one instance is positive, i.e. the highest score should be equal to the weak label, as written in the following relation for the i^{th} instance: $\max_j \hat{y}_{ikj} = y_{ik}$, where $\hat{y}_{ikj}$ and y_{ik} are the prediction for frame j and class k and the reference tag for class k, respectively.

There are drawbacks to this approach. For instance, the training of the model will focus only on the highest scored frame and ignore the other ones. To remedy to this issue, Morfi and Stowell [14] proposed a loss function that takes into account frames with the lowest prediction scores, which should tend towards zero, and also a naïve assumption that in general a specific event will be present in half of the frames. They applied this idea successfully on a binary classification problem, namely the presence/absence of singing birds in audio recordings. For the need of the present challenge, we generalized their loss function to $K > 2$ classes, as written in equation 3, where binCE stands for the binary cross-entropy loss. In our experiments, a first network is used to identify which classes should be considered for localization by a second network trained to minimize the MIL loss function.

$$\begin{aligned}\text{loss} = \sum_{k=1}^{K} \text{binCE}(y_{ik}, \max_j \hat{y}_{ikj}) + \\ \text{binCE}(y_{ik}/2, \text{mean}_j\, \hat{y}_{ikj}) + \text{binCE}(0, \min_j \hat{y}_{ikj})\end{aligned} \tag{3}$$

4. EXPERIMENTAL SETUP

4.1. Audio material

The DCASE 2018 Task 4 is related to discovering audio events from a set of 10 sound categories occurring in domestic environments, namely Speech, Dog, Cat, Alarm/Bell ringing, Dishes, Frying, Blender, Running water, Vacuum cleaner, and Electric shaver/toothbrush. All the files are 10-second clips extracted from Youtube user videos and are part of the Audioset corpus [2]. The recordings most often contain several overlapping event categories.

The challenge corpus is divided into three subsets: the training, test and evaluation subsets. Three different splits of training data were provided: a labeled, an unlabeled in domain and an unlabeled out of domain training sets. In our work, we only used the labeled

subset to train our models, and, in the case of GRU-WGRU only, we also tried to extend it by using pseudo-labels made on the unlabeled in domain subsets. The labeled training subset is comprised of 1578 clips (2244 class occurrences) for which weak annotations have been verified and cross-checked.

The test subset is comprised of 288 files for which we were provided with strong labels. We will thus report performance results on this subset. The results obtained on the evaluation subset, comprised of 880 files, are not available.

4.2. Audio features

For both approaches, only the labeled (*weak* labels) and the *unlabeled in-domain* subsets were used. We used the first one to train a classifier (for both WGRU & MIL), and also to add weak annotations to the *unlabeled in domain* subset. Finally, both subsets were used to retrain the model and perform localization. The use of the unlabeled in-domain subset proved useful for WGRU only, but not for MIL. More experiments are needed to draw conclusions for this semi-supervised setting.

As input to the networks, 64 log-Mel filter-bank (F-BANK) coefficients were extracted every 23 ms on 100 ms duration frames, with 20 Hz and 11025 Hz as minimum and maximum frequency values to compute the Mel bands, respectively. Hence, for each 10-second file, a 431×64 matrix is extracted. This matrix is used as a single input image fed to the networks.

Different normalization and features scaling methods were tested as pre-processing stage such as *global mean removal*, *mean and variance standardization*, but no gains were observed compared to using raw F-BANK.

4.3. Neural networks

For both approaches, we use two recurrent CNNs: the first for audio event classification at file level, and the second for localization. Both networks are identical to the baseline ones [18] except for WGRU, we use 2-d spatial dropout instead of standard dropout and 2×4 max-pooling instead of 1×4 for the first convolution layer. Spatial dropout allows to decrease correlation between activation maps and eventually overfitting.

For the MIL approach, in the first classification network, the GRU layer is replaced by a 2-d average- and a 2-d max- global pooling layer followed by a dense one with 1024 units. For classification, this network was found to perform better than the recurrent one. It yielded 85.84% and 82.86% f1 scores on our training and validation subset (proportion: 80/20 % of the weakly labeled training set).

Regarding the second network, used for localization, its architecture is the same as the baseline localization CRNN. The output of this network for a single recording is of dimension 431×10, 431 being the number of time frames, and 10 the number of classes.

In all cases, we used the Adam optimizer and a simple learning rate decay policy: dividing it by two after 30 epochs and 60 epochs. All the networks were trained on 100 epochs except the MIL localization network trained on 10 epochs only.

4.4. Threshold optimization

The score curves are individually rescaled to $[0, 1]$. The final event segments are obtained by first smoothing the score curves with a moving-average filter of size 19 frames. Second, the curves are binarized with a 0.07 threshold for MIL only. Neighbor segments are merged when separated by less than 200 ms, the tolerance margin used for evaluation.

For classification (WGRU and MIL) and localization (WGRU only), we use an ad-hoc threshold optimization algorithm to set the ten thresholds. It consists of a genetic algorithm inspired by simulated annealing [19]. This method reaches optimal values much faster than grid search.

5. RESULTS

Approach	Baseline	WGRU	MIL
F-score (%)	14.06	16.77	24.58
Alarm_bell_ringing	3.9	17.6	28.3
Blender	15.4	11.6	10.1
Cat	0.0	0.0	48.9
Dishes	0.0	0.0	0.0
Dog	0.0	4.8	18.6
Electric_shaver_toothbrush	32.4	33.3	28.6
Frying	31.0	29.5	26.7
Running_water	11.4	7.1	10.3
Speech	0.0	19.4	22.3
Vacuum_cleaner	46.5	40.0	52.2

Table 1: Global and class-wise F-measures (F-scores) on the test subset.

Table 1 shows the performance results on the test subset in terms of F-measure (F-score) for the baseline and our approaches. Both WGRU and MIL outperform the 14.06% baseline F-score with 16.34% and 24.58% scores, respectively. MIL behaves better than GRU-WGRU for all the classes.

	Weight(s)	F-score (%)	ER
Baseline	_	14.06	1.54
WGRU	1.	6.68	2.55
	0.50	4.69	2.92
	0.30	8.24	3.18
	0.20	11.35	3.37
Combined WGRUs	1. and 0.20	16.77	1.60

Table 2: Performance comparison and impact of the weight used with WGRU.

We only reported our best results in this table, using classification thresholds optimized on our validation subset (20% of the training data). Classification performance decreases by about 1% in F-score if using the default 0.5 threshold for all the classes.

5.1. WGRU

5.1.1. Temporal dependency weakening improves localization

Figure 2 gives an example in which the standard CRNN (GRU) does not allow to localize the sound event *Dog* that was correctly detected for that particular recording. The curves are the output of the last time-distributed layer of the network, the Green one being the curve using GRU and the orange one using WGRU. The vertical rectangles denote the ground truth and represent the segments where the dog should be detected. For GRU, *Dog* is detected throughout the

whole audio clip. For WGRU, with $\omega = 0.25$, the *dog* segments are properly localized based on the curve peaks.

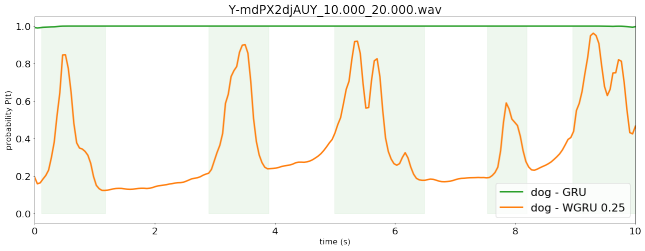


Figure 2: GRU (green) and WGRU (orange) score curves for the correct class *Dog*. The vertical rectangles denote the groundtruth and represent the segments where *Dog* should be detected. *Green* Localization results of the CRNN with GRU. The class *dog* is detected but in the entire clip. *Orange:* The prediction of the CRNN with WGRU with a temporal weight of 0.25. The dog barking segments are detected and localized properly.

This failure of the standard CRNN may be due to the fact that it is trained to detect a sound event in 10-s duration recordings, regardless of where in the file. Thus, the memory brought by the recurrent cell states keeps the information that *Dog* is present as soon as it is detected in the file, either at the beginning or at the end of the recording since bi-directional GRU layers are used.

5.1.2. *Combination of WGRUs*

Table 2 allows to compare the baseline performance of 14.06 in F-score and 1.54 in Error Rate (ER) to WGRU with different values of the weighting scalar ω. As one can see, WGRUs alone are worse than the baseline. The best weighting factor value was found to be estimated to about 0.20, with a 11.35% F-score. Smaller weight values revealed less efficient, showing that keeping some information from the previous hidden cell state is important. The best results, also reported in Table 1, were obtained by combining two WGRUs with 1. and 0.20 weights.

The combination of WGRU predictions is subjective and made after observation using the test dataset. The classes have been divided into two categories: stationary sounds (*Blender*, *Electric_shaver_toothbrush*, *Running_water*, *Vacuum_cleaner*) and short sounds (*Alarm_bell_ringing*, *Cat*, *Dog*, *Speech*). The predictions of the WGRU weighted at 1 are kept for the stationary sounds and the predictions of the WGRU weighted at 0.20 for the short sounds.

5.2. MIL

Figure 3 shows a successful example of the MIL model for a test file that contains speech and dog barking in segments given below the spectrogram. The first classification CNN correctly identified these two classes at file level, and the second MIL-CNN provides the two peaky curves, blue for *Speech*, red for *Dog*.

As shown in Table 1, MIL outperformed the baseline by a large margin for about haf of the classes and especially for *Cat* with a 48.9% F-score. For the other classes, its performance is lower but close. It is remarkable that all approaches failed for *Dishes*.

By observing the localization predictions, it appears that the MIL model confuses *Dishes* and *Frying*. In the training subset, about 46% of the Dishes samples also contain *Frying*. There are

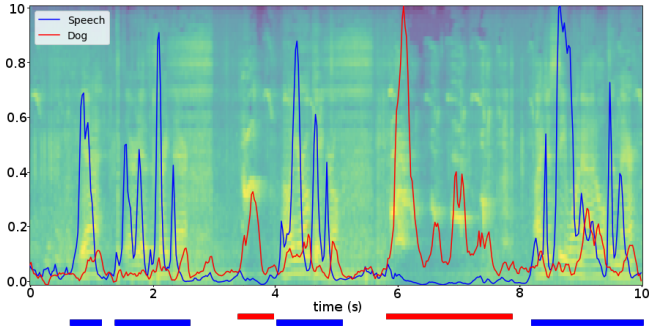


Figure 3: Score curves by MIL for the two correctly detected classes 'Speech' (Blue) and 'Dog' (Red). Below the spectrogram is represented the groundtruth.

even more files with *Dishes* and *Speech*, namely 52%, and 32% with the three classes together. *Dishes* is not confounded that much with *Speech* probably because there are many more Speech files than Dishes files: 550 versus 184 files.

6. CONCLUSION

In this paper, we reported experiments in the framework of the task four of the DCASE 2018 challenge. We had a two-fold objective of first, detecting sound events globally in audio recordings, second, localizing as precisely as possible where the detected event categories occur in time. We trained our models on weakly annotated dataset. The weak annotations provide tags from audio event but does not provide their temporal boundaries.

We explored two new approaches: 1) a "weighted-GRU" one (WGRU), in which we train a Convolutional Recurrent Neural Network for classification and then exploit its frame-based predictions at the output of the time-distributed dense layer to perform localization. We propose to lower the influence of the hidden states to avoid predicting a same score throughout a recording ; 2) An approach inspired by Multi-Instance Learning (MIL), in which we train a CNN to give predictions at frame-level, using a custom loss function based on the weak label and statistics of the frame-based predictions. Both approaches outperform the baseline of 14.06% in F-measure by a large margin, respectively 16.34% and 24.58% for combined WGRUs and MIL, on a test set comprised of 288 files.

Limitations were described. In particular, when two classes of sound events occur very often together, such as Dishes and Frying, MIL fails to learn how to distinguish between them. Our next step will be to modify the MIL loss function to penalize the fact that the prediction outputs for two different classes are too similar. Another improvement would be to achieve the same performance but using a single neural network rather than two.

7. ACKNOWLEDGMENT

This work is partially supported by the Labex ANR-10-LABX-0002-01 (ANR-5-IDEX-002, AMIES-UMS 3458), within the PEPS REP4SPEECH project, and the LUDAU ANR project.

We thank Benjamin Chamand for his valuable suggestions and technical support, and Julien Pinquier for his advises on acoustic features.

References

[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[2] J. F. Gemmeke, D. P. Ellis, D. Freedman, A. Jansen, W. Lawrence, R. C. Moore, M. Plakal, and M. Ritter, "Audio set: An ontology and human-labeled dataset for audio events," in *Proc. ICASSP*. IEEE, 2017, pp. 776–780.

[3] R. Serizel, N. Turpault, H. Eghbal-Zadeh, and A. Parag Shah, "Large-Scale Weakly Labeled Semi-Supervised Sound Event Detection in Domestic Environments," July 2018, submitted to DCASE2018 Workshop. [Online]. Available: https://hal.inria.fr/hal-01850270

[4] T. Virtanen, M. D. Plumbley, and D. Ellis, *Computational analysis of sound scenes and events*. Springer, 2018.

[5] D. Stowell, M. Wood, Y. Stylianou, and H. Glotin, "Bird detection in audio: a survey and a challenge," *arXiv preprint arXiv:1608.03417*, 2016.

[6] M. Crocco, M. Cristani, A. Trucco, and V. Murino, "Audio surveillance: A systematic review," *ACM Comput. Surv.*, vol. 48, no. 4, pp. 52:1–52:46, Feb. 2016. [Online]. Available: http://doi.acm.org/10.1145/2871183

[7] P. Guyot, J. Pinquier, and R. André-Obrecht, "Water sound recognition based on physical models," in *Proc. ICASSP*. IEEE, 2013, pp. 793–797.

[8] E. Cakir, T. Heittola, H. Huttunen, and T. Virtanen, "Polyphonic sound event detection using multi label deep neural networks," in *Neural Networks (IJCNN), 2015 International Joint Conference on*. IEEE, 2015, pp. 1–7.

[9] H. Zhang, I. McLoughlin, and Y. Song, "Robust sound event recognition using convolutional neural networks," in *Proc. ICASSP*. IEEE, 2015, pp. 559–563.

[10] G. Parascandolo, H. Huttunen, and T. Virtanen, "Recurrent neural networks for polyphonic sound event detection in real life recordings," *arXiv preprint arXiv:1604.00861*, 2016.

[11] E. Cakir, G. Parascandolo, T. Heittola, H. Huttunen, T. Virtanen, E. Cakir, G. Parascandolo, T. Heittola, H. Huttunen, and T. Virtanen, "Convolutional recurrent neural networks for polyphonic sound event detection," *Proc. TASLP*, vol. 25, no. 6, pp. 1291–1303, 2017.

[12] A. Kumar and B. Raj, "Audio event detection using weakly labeled data," in *Proceedings of the 2016 ACM on Multimedia Conference*. ACM, 2016, pp. 1038–1047.

[13] ——, "Deep cnn framework for audio event recognition using weakly labeled web data," *arXiv preprint arXiv:1707.02530*, 2017.

[14] V. Morfi and D. Stowell, "Data-efficient weakly supervised learning for low-resource audio event detection using deep learning," *arXiv preprint arXiv:1807.06972*, 2018.

[15] R. Dey and F. M. Salem, "Gate-variants of gated recurrent unit (GRU) neural networks," *CoRR*, vol. abs/1701.05923, 2017. [Online]. Available: http://arxiv.org/abs/1701.05923

[16] K. Cho, B. van Merriënboer, Ç. Gülçehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder–decoder for statistical machine translation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1724–1734. [Online]. Available: http://www.aclweb.org/anthology/D14-1179

[17] T. G. Dietterich, R. H. Lathrop, and T. Lozano-Pérez, "Solving the multiple instance problem with axis-parallel rectangles," *Artificial intelligence*, vol. 89, no. 1-2, pp. 31–71, 1997.

[18] http://dcase.community/workshop2018/.

[19] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by Simulated Annealing," *Science*, vol. 220, pp. 671–680, May 1983.

GENERAL-PURPOSE TAGGING OF FREESOUND AUDIO WITH AUDIOSET LABELS: TASK DESCRIPTION, DATASET, AND BASELINE

Eduardo Fonseca[1]*, *Manoj Plakal*[2], *Frederic Font*[1], *Daniel P. W. Ellis*[2], *Xavier Favory*[1], *Jordi Pons*[1], *Xavier Serra*[1]

[1]Music Technology Group, Universitat Pompeu Fabra, Barcelona {name.surname}@upf.edu
[2] Google, Inc., New York, NY, USA {plakal,dpwe}@google.com

ABSTRACT

This paper describes Task 2 of the DCASE 2018 Challenge, titled "General-purpose audio tagging of Freesound content with AudioSet labels". This task was hosted on the Kaggle platform as "Freesound General-Purpose Audio Tagging Challenge". The goal of the task is to build an audio tagging system that can recognize the category of an audio clip from a subset of 41 diverse categories drawn from the AudioSet Ontology. We present the task, the dataset prepared for the competition, and a baseline system.

Index Terms— Audio tagging, audio dataset, data collection

1. INTRODUCTION

The sounds in our everyday environment carry a huge amount of information of the events occurring nearby. Humans are able to recognize and discern many sound events but state-of-the-art automatic processing of sounds by machines still lags far behind. Further research is needed to develop robust systems capable of recognizing a wide range of sound events in realistic audio streams [1]. In recent years, the various editions of the DCASE Challenge have provided scenarios for evaluating different computational methods for several sound recognition tasks using common publicly available datasets [2]. This paper describes the characteristics, dataset and baseline of DCASE 2018 Task 2 "General-purpose audio tagging of Freesound content with AudioSet labels".

There have been two audio tagging tasks in previous DCASE editions, each focused on a specific domain of sounds. In DCASE 2016 [3], the task targeted domestic audio tagging for which the CHiME-Home dataset was used, including 7 sound categories and 6.8h of recordings [4]. In DCASE 2017 [2], the task focused on audio tagging in the context of smart cars, for which a large-scale dataset featuring 17 categories was utilized. Other popular datasets for sound event classification are the ESC-50 [5] and the UrbanSound8k [6] datasets. The former counts with 50 diverse categories but less than 3h of audio recordings. The latter is designed to enable urban sound research, featuring 10 categories and almost 9h of audio. Hence, many of available data resources for sound event classification are domain-specific, and/or of relatively small size. Recently, however, general-purpose sound event recognizers have gained attention, where a wide range of sounds events are considered, not tied to a specific domain. This research has been mostly triggered by AudioSet, a large-scale audio dataset structured with an ontology of 632 sound events [7].

*This work is partially supported by the European Union's Horizon 2020 research and innovation programme under grant agreement No 688382 AudioCommons and a Google Faculty Research Award 2017.

In this paper, we focus on general-purpose audio tagging using a dataset of 41 categories and almost 18h of training data. Specifically, the goal of this task is to build an audio tagging system that can categorize an audio clip as belonging to one of a set of 41 diverse categories drawn from the AudioSet Ontology (related to musical instruments, human sounds, domestic sounds, animals, etc.). One of the motivations for this task comes from the large amount of user-generated audio content that is available on the web, which can be a resource of great potential for sound recognition related research. The use of such data for training audio tagging systems poses issues that have not been addressed in previous DCASE Challenges. In particular, this task deals with user-generated audio clips retrieved from Freesound,[1] which are very diverse in terms of acoustic content, recording techniques, clip duration, etc. Likewise, these audio clips sometimes feature incomplete and inconsistent user-provided metadata. To prepare the dataset for this task, some audio clips were manually labeled using the subset of 41 categories, while a larger set of clips was automatically categorized on the basis of their existing user-provided metadata (see Section 3 for more details). As a result, the dataset features a small amount of reliable annotations, and a large amount of non-verified annotations that could include a small amount of label noise.

Therefore, this task addresses two main challenges of *i)* recognizing an increased number of diverse sound events, and *ii)* leveraging subsets of training data featuring annotations of varying reliability. Submissions to this task will provide insight towards the development of broadly-applicable sound event classifiers. Potential applications include automatic description of multimedia content, and acoustic monitoring applications. This paper is organized as follows. Section 2 provides more details about the task and its experimental setup. Section 3 presents in detail the dataset prepared for the task, and Section 4 describes a baseline system. Final remarks are given in Section 5.

2. TASK SETUP

The goal of this task is to predict the category for each audio clip in a test set. The task setup is a multiclass classification problem, and hence the systems to be developed in this task can be denoted as *single-tag* audio tagging systems, as illustrated in Fig. 1. This task was hosted on Kaggle—a platform for machine learning competitions—and ran from March 30th to July 31st 2018. The resources associated to this task (dataset download, submission, and leaderboard) can be found on the Kaggle competition page.[2]

[1] `https://freesound.org`

[2] `https://kaggle.com/c/freesound-audio-tagging`
Note that competition name on Kaggle is abbreviated from the full DCASE task name to "Freesound General-Purpose Audio Tagging Challenge".

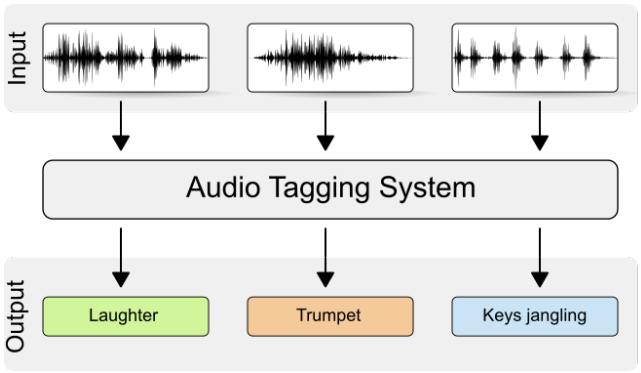


Figure 1: Overview of a *single-tag* tagging system.

As described in Section 3, the audio data for this task are split into a train set and a test set, both made publicly available when the competition launched. The train set, for which ground-truth annotations were provided, is used for system development while the test set is kept for the evaluation of the resulting systems. The test set, whose true labels were not released, is further divided into two divisions: *i)* 19% of the test samples are used to calculate the *public* leaderboard (providing a live ranking of all participants), and *ii)* the remaining 81% feeds the *private* leaderboard, used for the final ranking which is revealed only when the competition ends.

2.1. Evaluation Metric and Competition Rules

The task used mean Average Precision @ 3 (mAP@3) as the evaluation metric, as defined in the Evaluation section of the competition page.[2] This metric accepts up to three ranked predicted labels for each audio clip in the test set, and gives full credit if the correct label occurs first, with lesser credit for correct label predictions in second or third place.

Participants were required to run their systems on the test set and submit the system output—the predicted labels—in a comma-separated data file (CSV). Participants could submit a maximum of two submissions per day, and select two *final* submissions to be considered for the ranking. Additionally, participants were encouraged to submit a technical report describing their systems. A detailed description of the task rules can be found in the Rules section of the competition page,[2] and the most important points are summarized in the DCASE Challenge page.[3]

2.2. Judges' Award

To complement the leaderboard results of the mAP-based ranking, the organizers of the task introduced a complementary Judges' Award to promote submissions using novel, problem-specific and efficient approaches. Details about the Judges' Award rules and requirements can be found in the Discussion section of the competition page.[4]

3. DATASET

The dataset used for the task was prepared by the task organizers during the months previous to the start of the competition, and is called "Freesound Dataset Kaggle 2018" (or *FSDKaggle2018* for short). FSDKaggle2018 is in fact a reduced subset of *FSD*, which is a large-scale, general-purpose open audio dataset that is currently under development. More details about FSD can be found in [8]. The following subsections describe the creation process of FSDKaggle2018.

3.1. Source of Audio Content

FSDKaggle2018 is composed of audio content collected from Freesound. Freesound is a sound sharing site developed and maintained by the Music Technology Group [9]. At the time of writing, Freesound hosts more than 380,000 sounds uploaded by its community of users. Freesound content is very heterogeneous, including sounds from a wide range of real-world environments, from musical and human-generated sounds to animal sounds or artificially generated sound effects. The authors of the sounds uploaded to Freesound are asked to provide some basic metadata (e.g., tags, title and textual descriptions). This metadata is then used for searching and browsing and is also very valuable for research purposes. All the content in Freesound is released under Creative Commons licenses which facilitate its sharing and reuse. Since sounds are uploaded by thousands of users across the globe, recording scenarios and techniques can vary widely. We hypothesize that this fact makes Freesound content representative of real-world situations.

3.2. Annotation Procedure

FSDKaggle2018 is organized using categories of the AudioSet Ontology. As a first step, we did a mapping of 268,261 Freesound clips to their corresponding AudioSet categories. To do that, we assigned a number of Freesound tags to almost all of the 632 AudioSet categories and, for each category, we selected audio clips from Freesound tagged with at least one of these tags. This process led to a number of automatically generated *candidate annotations* that express the potential presence of a sound category in an audio clip. These annotations are at the clip level and hence can be considered weak labels. However, some audio files are specific sound examples of the category under consideration, where the acoustic signal fills almost the entirety of the file, which could arguably be considered as strong labels.

In order to validate the candidate annotations, we used *Freesound Datasets*,[5] an online platform for the collaborative creation of open audio datasets developed at the Music Technology Group. We deployed a validation task in which Freesound Datasets users can manually verify the presence or absence of a candidate sound category in an audio clip with a *rating* mechanism. For every sound category, users first go through a training phase to get familiar with the category, read its description provided by AudioSet, and listen to some selected sound examples. Then, users are presented with a series of audio clips, and prompted the question: *Is <category> present in the following sounds?* Users must select one of the response types listed in Table 1. Along with the audio clips, users are also given links to the corresponding Freesound sound pages where the original tags and descriptions are available and can be used as an aid for the validation process. Participants in the validation task included voluntaries from the Freesound community as well as researchers and students from the Music Technology Group.

Among the various features implemented in the validation task, it is worth mentioning the utilization of quality control mechanisms

[3] http://dcase.community/challenge2018/task-general-purpose-audio-tagging#task-rules

[4] https://www.kaggle.com/c/freesound-audio-tagging/discussion/59932

[5] https://datasets.freesound.org

such as the periodic inclusion of verification clips to test the reliability of the submitted responses. Likewise, in order to choose which audio clips to present to each user, we adopt a prioritization scheme that considers inter-annotator agreement. More specifically, each candidate annotation is presented to several users (i.e., annotators) until agreement is attained by two different users on a response type. When a candidate annotation reaches an agreement status, it is considered validated and is no longer presented to other users.

3.3. Dataset Curation

After generating candidate annotations and collecting user ratings in the Freesound Datasets platform, each candidate annotation had a particular distribution of ratings {PP, PNP, NP, U} (see Table 1). Then, a curation step was carried out to select which categories and audio clips to be finally included in FSDKaggle2018. Considering all annotations, two annotation subsets were created for each sound category:

- **Manually-verified annotations**: composed of those annotations rated only as PP (a great majority with inter-annotator agreement but not all of them, hence PP & PP or single PP).
- **Non-verified annotations**: composed mainly of the *un-rated* candidate annotations, and complemented with a small amount of rated annotations. This small amount of rated annotations can include any rating distribution except *i)* those corresponding to the manually-verified portion, and *ii)* those that clearly denote an incorrect mapping (e.g., NP, NP & U, etc.).

For each sound category, a quality estimate *QE* for the non-verified annotations can be computed according to (1)

$$QE = \frac{\#PP + \#PNP}{\#PP + \#PNP + \#NP + \#U} \quad (1)$$

where $\#X$ denotes the number of ratings of type X gathered in the category.

Next, a number of restrictions were applied sequentially to the categories and/or the audio clips within them. First, we discarded all categories not belonging to leaf nodes of the AudioSet hierarchy, leaving a total of 474 categories. Then, we removed audio clips shorter than 300ms and longer than 30s, as well as those clips with Creative Commons *Non-commercial* or *Sampling+* licenses. All sound categories that, after the previous filtering, did not have *i)* a minimum of number of manually-verified annotations, and *ii)* a minimum number of manually-verified + non-verified annotations, were discarded. Note that in order to accept the non-verified annotations in a category, a minimum QE was required (see Section 3.4).

We observed that quite a few leaf sound categories were discarded because they did not have sufficient number of clips. In some of these cases, making use of the hierarchical relationships in AudioSet, we decided to aggregate the content of these leaf categories together with that of their immediate parents in order to create new candidate parent categories. Similar requirements (in terms of QE and amount of data) were applied to these newly formed categories for them to be accepted in the raw version of FSDKaggle2018.

After this process, an analysis was carried out in terms of *i)* number of *in-domain*[6] candidate annotations per audio clip and *ii)* semantic aspect of the resulting categories. The analysis revealed that the vast majority of the audio clips presented a single candidate annotation and, for the sake of simplicity, we decided to discard audio clips with multiple annotations.[7] We also discarded a few categories with somewhat abstract or vague meaning like "Recording" or "Effect unit".

Finally, the audio clips with manually-verified annotations for every category were split into roughly 70%/30% for train and test sets. The split was carried out considering, whenever possible, clip origin (by using part of the Freesound metadata) and clip duration (so as to have short and long clips in both sets). Then, we complemented the manually-verified portion of the train set with the non-verified annotations. This addition was performed such that the maximum number of clips per category was 300 in order to mitigate data imbalance among categories. The dataset curation resulted in the selected 11,073 sounds/annotations organized with 41 AudioSet categories.

3.4. Dataset Description

FSDKaggle2018 contains a total of 11,073 files provided as uncompressed PCM 16 bit, 44.1 kHz, mono audio files. All audio clips are released under either Creative Commons *Attribution* or *Zero* licenses. The clips are unequally distributed in the 41 categories of the AudioSet Ontology listed in Table 2. The dataset most relevant characteristics are as follows:

- Audio clips are annotated with a single ground truth label.
- The duration of the audio clips ranges from 300ms to 30s due to the diversity of the sound categories and the preferences of Freesound users when recording sounds.
- The dataset is split into a train set and a test set.
- The **train set** is meant to be for system development and includes 9473 audio clips unequally distributed among 41 categories. The minimum number of audio clips per category in the train set is 94, and the maximum is 300. The total duration of the train set is almost 18h.
- Out of the 9473 clips from the train set, 3710 have manually-verified annotations and 5763 have non-verified annotations. The latter are properly flagged so that participants can opt to use this information during the development of their systems.
- The **test set** is composed of 1600 clips with manually-verified annotations and with a similar category distribution to that of the manually-verified portion of the train set. The minimum number of manually-verified audio clips per category in the test set is 25, and the maximum is 110. The test set is complemented with 7800 *padding* clips.[8] These clips, which are not used for scoring the systems, are added to prevent undesired practices (considering that the test set was made publicly available when the competition launched).

As mentioned in Section 3.3, all **manually-verified annotations** are annotations validated as PP (Present and Predominant). This means that, in most cases, there is no additional acoustic material other than the labeled category. In few cases, there may be some additional sound events, but these additional events will be *out-of-domain*, i.e., they won't belong to any of the 41 AudioSet categories of FSDKaggle2018. The **non-verified annotations** have a QE of *at least* 65% in each category. This means that some of them are most

[6] Considering only the set of valid categories at this point of the process, instead of all the AudioSet categories.

[7] Note that the automatically generated candidate annotations depend on the user-generated tags in Freesound and on the mapping to the AudioSet Ontology. Hence their reliability relies on the subsequent validation process.

[8] Hence, the dataset available from Kaggle contains 18,873 audio files.

probably inaccurate. It can happen that audio clips corresponding to some of the non-verified annotations present several sound sources (even though only one label is provided as ground truth). These additional sources are typically out-of-domain, but in a few cases they could be within the domain. Fig. 2 shows the distribution of manually-verified and non-verified annotations per category in the train set.

4. BASELINE SYSTEM

In recent years, convolutional and recurrent neural networks (CNNs, RNNs, CRNNs) have achieved state-of-the-art performance for audio tagging and event detection in DCASE Challenges [10, 11], and deep CNNs have been shown to work well with very large datasets [12, 13], outpacing simpler models.

Our baseline uses a relatively shallow 3-layer CNN with log-mel spectrogram input features and a 41-way softmax classifier layer, described in detail in the public release [9]. Incoming audio (always 44.1 kHz mono) is divided into overlapping windows of size 0.25s with a hop of 0.125s. These windows are decomposed with a short-time Fourier transform using 25ms windows every 10ms. The resulting spectrogram is mapped into 64 mel-spaced frequency bins, and the magnitude of each bin is log-transformed after adding a small offset to avoid numerical issues. The model consists of three 2-D convolutional layers and alternating max-pooling layers, followed by a softmax classifier layer. Predictions are obtained for a clip of arbitrary length by running the model over 0.25s-wide windows every 0.125s, and averaging all the window-level predictions to obtain a clip-level prediction.

The baseline achieves an mAP@3 of 0.70 on the entire test set (0.70 and 0.69 on the public and private Kaggle leaderboard splits, respectively) after training for 5 epochs on the train set. Per-category AP@3 is reported in Table 2.

5. CONCLUSION

In this paper we have described the task setup, dataset, and baseline of DCASE 2018 Task 2 "General-purpose audio tagging of Freesound content with AudioSet labels". This task was hosted on the Kaggle platform as "Freesound General-Purpose Audio Tagging Challenge" and ran from March 30th to July 31st 2018. The main focus of the paper is the description of FSDKaggle2018, the dataset we prepared for the task. FSDKaggle2018 presents the particularities of having subsets of training data with annotations of varying reliability as well as featuring variable-length audio clips, both novel challenges in DCASE competitions. The dataset is currently available on the Kaggle competition page, and future updates of the dataset (including ground-truth data for the test set and extra associated Freesound metadata) will be made publicly available in the Freesound Datasets platform. Through FSDKaggle2018 and the provided baseline system, this competition intends to foster open research in sound event recognition.

6. ACKNOWLEDGMENT

We thank Addison Howard and Walter Reade of Kaggle for their invaluable assistance with the task design and Kaggle platform, and everyone who contributed to FSDKaggle2018 with annotations. Eduardo Fonseca is also grateful for the GPU donated by NVidia.

[9] https://github.com/DCASE-REPO/dcase2018_baseline/tree/master/task2

7. REFERENCES

[1] T. Virtanen, M. D. Plumbley, and D. Ellis, *Computational Analysis of Sound Scenes and Events*. Springer, 2018.

[2] A. Mesaros, T. Heittola, A. Diment, B. Elizalde, A. Shah, E. Vincent, B. Raj, and T. Virtanen, "DCASE 2017 challenge setup: Tasks, datasets and baseline system," in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2017 Workshop (DCASE2017)*, November 2017, submitted.

[3] A. Mesaros, T. Heittola, E. Benetos, P. Foster, M. Lagrange, T. Virtanen, and M. D. Plumbley, "Detection and classification of acoustic scenes and events: Outcome of the dcase 2016 challenge," *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, vol. 26, no. 2, pp. 379–393, 2018.

[4] P. Foster, S. Sigtia, S. Krstulovic, J. Barker, and M. D. Plumbley, "Chime-home: A dataset for sound source recognition in a domestic environment," in *Workshop on Applications of Signal Processing to Audio and Acoustics*. IEEE, 2015, pp. 1–5.

[5] K. J. Piczak, "Esc: Dataset for environmental sound classification," in *Proceedings of the ACM International Conference on Multimedia*. ACM, 2015, pp. 1015–1018.

[6] J. Salamon, C. Jacoby, and J. P. Bello, "A dataset and taxonomy for urban sound research," in *Proceedings of the ACM International Conference on Multimedia*. ACM, 2014, pp. 1041–1044.

[7] J. F. Gemmeke, D. P. W. Ellis, D. Freedman, A. Jansen, W. Lawrence, R. C. Moore, M. Plakal, and M. Ritter, "Audio set: An ontology and human-labeled dataset for audio events," in *Proc. IEEE ICASSP 2017*, New Orleans, LA, 2017.

[8] E. Fonseca, J. Pons, X. Favory, F. Font, D. Bogdanov, A. Ferraro, S. Oramas, A. Porter, and X. Serra, "Freesound datasets: a platform for the creation of open audio datasets," in *Proceedings of the 18th International Society for Music Information Retrieval Conference (ISMIR 2017)*, Suzhou, China, 2017, pp. 486–493.

[9] F. Font, G. Roma, and X. Serra, "Freesound technical demo," in *Proceedings of the ACM International Conference on Multimedia*. ACM, 2013, pp. 411–412.

[10] http://www.cs.tut.fi/sgn/arg/dcase2016/task-results-audio-tagging.

[11] http://www.cs.tut.fi/sgn/arg/dcase2017/challenge/task-large-scale-sound-event-detection-results.

[12] S. Hershey, S. Chaudhuri, D. P. W. Ellis, J. F. Gemmeke, A. Jansen, C. Moore, M. Plakal, D. Platt, R. A. Saurous, B. Seybold, M. Slaney, R. Weiss, and K. Wilson, "CNN architectures for large-scale audio classification," in *Proc. IEEE ICASSP 2017*, New Orleans, LA, 2017. [Online]. Available: https://arxiv.org/abs/1609.09430

[13] Y. Aytar, C. Vondrick, and A. Torralba, "Soundnet: Learning sound representations from unlabeled video," in *Advances in Neural Information Processing Systems*, 2016.

Table 1: Response types for the validation task. Users must select one to answer the question: *Is <category> present in the following sounds?*

Response type	**Meaning**
Present and predominant (PP)	The type of sound described is **clearly present** and **predominant**. This means there are no other types of sound, with the exception of low/mild background noise.
Present but not predominant (PNP)	The type of sound described is **present**, but the audio clip also **contains other salient types of sound and/or strong background noise**.
Not Present (NP)	The type of sound described is **not present** in the audio clip.
Unsure (U)	**I am not sure** whether the type of sound described is present or not.

Table 2: Categories composing FSDKaggle2018, along with the number of clips and time (in minutes, rounded) in the train set. Per-category AP@3 achieved by the baseline system is reported using all the test files for every category (i.e., not following the public/private splits of the Kaggle leaderboard).

Name	**clips**	**time**	**AP@3**	**Name**	**clips**	**time**	**AP@3**	**Name**	**clips**	**time**	**AP@3**
Acoustic guitar	300	52	0.67	Electric piano	150	25	0.75	Microwave oven	146	25	0.56
Applause	300	58	0.98	Fart	300	18	0.65	Oboe	299	15	0.88
Bark	239	45	0.85	Finger snapping	117	6	0.71	Saxophone	300	34	0.84
Bass drum	300	13	0.55	Fireworks	300	48	0.61	Scissors	95	16	0.37
Burping,eructation	210	12	0.71	Flute	300	46	0.90	Shatter	300	26	0.70
Bus	109	28	0.53	Glockenspiel	94	8	0.59	Snare drum	300	18	0.30
Cello	300	37	0.86	Gong	292	42	0.81	Squeak	300	38	0.16
Chime	115	24	0.79	Gunshot,gunfire	147	11	0.16	Tambourine	221	10	0.78
Clarinet	300	35	0.96	Harmonica	165	19	0.86	Tearing	300	39	0.94
Computer keyboard	119	23	0.54	Hi-hat	300	19	0.53	Telephone	120	16	0.65
Cough	243	22	0.69	Keys jangling	139	19	0.76	Trumpet	300	28	0.84
Cowbell	191	11	0.58	Knock	279	19	0.89	Violin,fiddle	300	27	0.73
Double bass	300	17	0.69	Laughter	300	36	0.96	Writing	270	48	0.66
Drawer open,close	158	18	0.05	Meow	155	19	0.82				

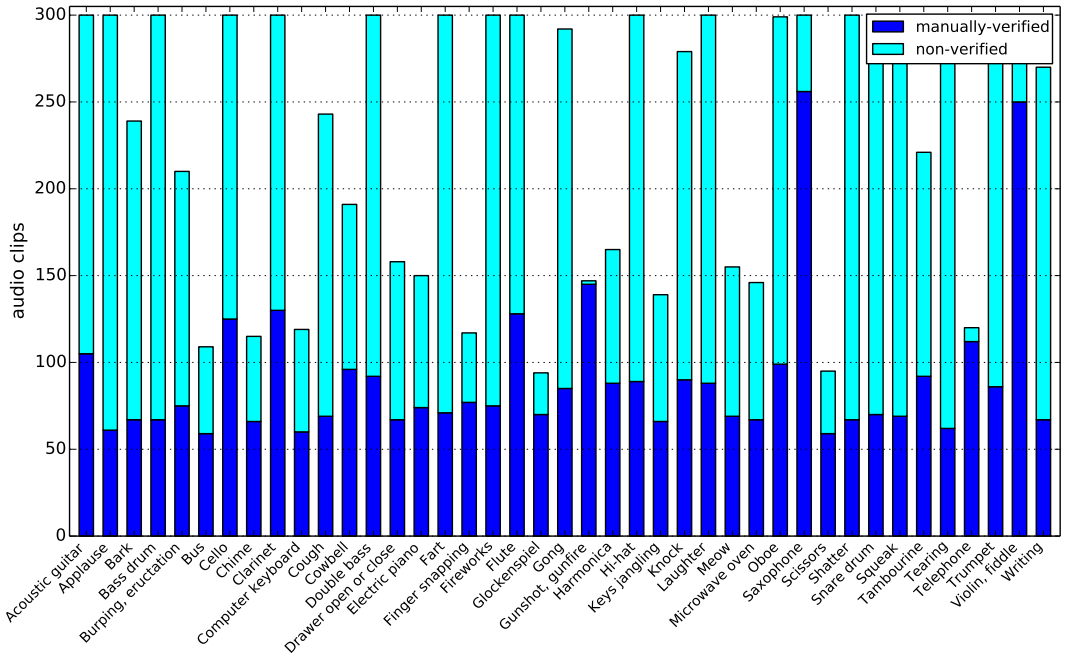


Figure 2: Distribution of manually-verified and non-verified annotations per category in the train set.

WEAKLY LABELED SEMI-SUPERVISED SOUND EVENT DETECTION USING CRNN WITH INCEPTION MODULE

Wootaek Lim, Sangwon Suh, Youngho Jeong

Realistic AV Research Group
Electronics and Telecommunications Research Institute
218 Gajeong-ro, Yuseong-gu, Daejeon, Korea
wtlim@etri.re.kr

ABSTRACT

In this paper, we present a method for large-scale detection of sound events using small weakly labeled data proposed in the Detection and Classification of Acoustic Scenes and Events (DCASE) 2018 challenge Task 4. To perform this task, we adopted the convolutional neural network (CNN) and gated recurrent unit (GRU) based bidirectional recurrent neural network (RNN) as our proposed system. In addition, we proposed the Inception module for handling various receptive fields at once in each CNN layer. We also applied the data augmentation method to solve the labeled data shortage problem and applied the event activity detection method for strong label learning. By applying the proposed method to a weakly labeled semi-supervised sound event detection, it was verified that the proposed system provides better performance compared to the DCASE 2018 baseline system.

Index Terms— DCASE 2018, Sound event detection, Weakly labeled semi-supervised learning, Deep learning, Inception module

1. INTRODUCTION

In the field of machine learning, there are various tasks for modeling human auditory cognitive systems. One such field is sound event detection (SED), which is a rapidly growing field owing to the improvement of algorithms, datasets, and expansion of smart devices [1, 2]. In particular, SED technology is being studied to provide services that inform people about the context information of sound events at home or outside. Moreover, SED is important for the auto-tagging of multimedia content [1, 2]. To contribute to the SED task, the DCASE challenge has been organized for three years beginning in 2013 [1, 2, 3]. This year, the DCASE 2018 challenge comprises five tasks: acoustic scene classification, general-purpose audio tagging of Freesound content with AudioSet labels, bird audio detection, large-scale weakly labeled semi-supervised SED in domestic environments, and monitoring of domestic activities based on multi-channel acoustics [4]. Among them, this paper describes a method for performing the fourth task of the DCASE 2018 challenge, large-scale detection of sound events using weakly labeled data. The goal of task 4 is to find the onset and offset of a sound event using the weak label in an audio clip. A variety of methods were proposed in the previous DCASE 2017 challenge [5, 6, 7, 8, 9] to solve this problem. Furthermore, a baseline system that performs the task is provided in the DCASE 2018 challenge [10]. Based on these previous studies, we propose a network with the Inception module and several ways to improve the performance. The remainder of this paper is organized as follows: Section 2 introduces the proposed network architecture for the weakly labeled semi-supervised SED; Section 3 presents the experimental settings and results using the DCASE 2018 dataset; and Section 4 draws the conclusions of our paper.

2. PROPOSED METHOD

We propose a weakly labeled semi-supervised SED method that uses the Inception architecture. To be specific, a CNN layer is implemented with the Inception structure and time information is learned using a bidirectional GRU. To perform SED, two separate learning stages are proposed. The first stage is for sound tagging and the second stage is for sound event detection.

2.1. Data augmentation

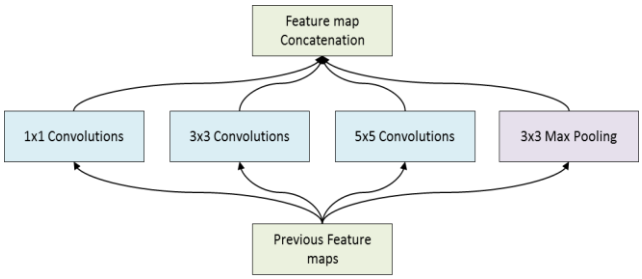
The fourth task of the DCASE 2018 challenge focuses on large-scale detection of sound events using small weakly labeled data. The challenge of this task is to explore the possibilities of leveraging large amounts of unbalanced and unclassified training data with a small set of annotated training data to improve system performance. As the weakly labeled data which is provided by DCASE challenge Task 4 is small, data augmentation is required to learn a better network. Data augmentation is the process of creating new training samples by making small changes to the original training data while keeping its characteristics. By performing the data augmentation, the network can be learned to improve its generalization ability for various unseen data [11]. Table 1 shows the data augmentation methods and details that we applied. To increase the performance of the classifier, we applied pitch shift manipulations with rates of 0.8, 0.9, 1.1, and 1.2. Moreover, the audio signal was stretched to 1.1 and 1.2 times faster and flipped horizontally to obtain a reversed image of the data.

Table 1: Data Augmentation methods and details.

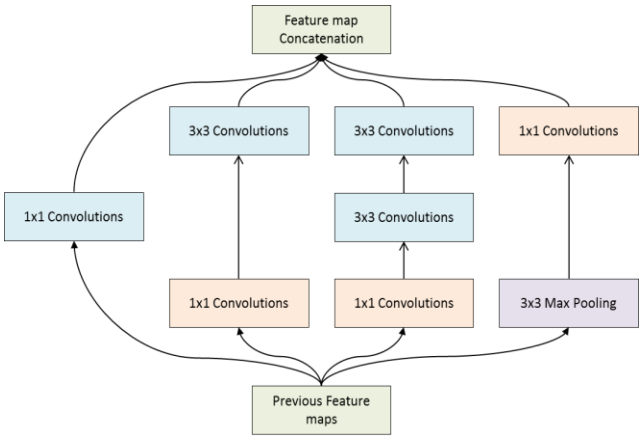
Data Augmentation Method	Value
Pitch Shift	0.8, 0.9, 1.1, 1.2
Time Stretch	1.1, 1.2
Reverse	Horizontal flip

2.2. Inception module

In CNN, each convolution filter learns a local part of an image or feature map. In other words, it is a combination of information in the local receptive field. This is accomplished by passing these combinations through the activation function to infer non-linear relationships and make larger features smaller, such as by pooling. Therefore, it is important to see the various receptive fields in one convolution layer. In this regard, the Inception architecture has been proposed in [12, 13, 14]. The key concept of the Inception architecture is based upon finding the optimal local sparse structure in a convolutional vision network that can be approximated and applied. Intuitively, visual information must be processed at various scales and aggregated to abstract features of different scales at the same time. Figure 1 shows the scheme of the naïve Inception module and modified Inception module with dimension reductions.



(a) Naïve Inception module



(b) Modified Inception module with dimension reductions

Figure 1: The scheme of the Inception module.

2.3. Proposed architecture

Figure 2 presents the network structure of the Inception architecture employed in our proposed system. It is a convolutional recurrent neural network (CRNN) structure that combines CNN and RNN. In this system, the audio signal is first converted to 64 log mel-band energies to form an input vector to the network. Next, the stem layer for extracting low-level features is applied using 3 × 3 convolution filters. After which, these extracted low-level features are used to train various receptive fields at once through the Inception layers. The Inception layers in Figure 2 correspond to those in Figure 1-(b). Moreover, max pooling is performed with the Inception layer to compress the information in the frequency axis. The recurrent layer is then stacked using a bidirectional GRU to learn the relevance between time frames and connect the dense connection in every single frame. The final result is output through a global average pooling (GAP) layer.

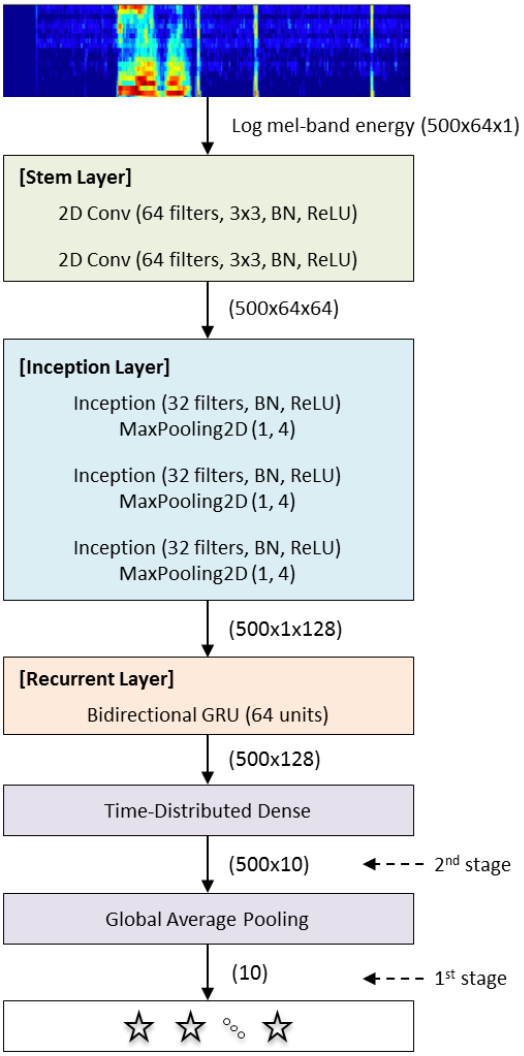


Figure 2: The structure of the proposed convolutional recurrent neural network for sound event detection with the Inception module.

2.4. First stage: sound tagging

In the first stage of the proposed system, a weak sound tagging network is trained using the weakly labeled data to assign predicted labels automatically to unlabeled in domain data. These unlabeled in domain data are excerpted from YouTube clips. To train the model, we use 80% of the weakly labeled data provided for the fourth task of the DCASE challenge as a training set; the rest of the data are used as a validation set. We apply the data augmentation method which is mentioned in section 2.1 to weakly labeled dataset to learn the generalized model.

2.5. Second stage: sound event detection

In the second stage, the SED network is trained using auto-tagged unlabeled in domain data which is automatically labeled in the first stage. This step is performed by excluding the GAP layer from the first stage network and outputting the frame recognition result. A simple method for learning strong label from weak label is to assign a strong label to all time frames. However, assigning strong labels from a weak label is difficult because it is impossible to know which frame has events or not. Owing to the absence of prior knowledge regarding the existence of the event, we calculate the log mel-band energies and assign a pseudo strong label when the average value of the log mel-band energies in each frame is above zero. This method assumes that there is no acoustic event in a frame where the energy is small. This is described as event activity detection in section 3.

3. PERFORMANCE EVALUATION

3.1. Pre-processing

To perform the experiment, we resampled the audio signal to a 16-kHz sampling rate and down-mixed it to a mono channel. The audio signal was then converted to 64 log mel-band energies with a frame size of 40 ms and an overlap length of 20 ms. At this time, the area of the mel-scale filter bank was normalized. As a result, we obtained an image with 500 time frames and 64 frequency bands and used it as a network input.

3.2. Hyperparameters and settings

In Table 2, we list the hyperparameters and settings used in this study. The number of nodes for each convolution layer was 64 or 32, and the convolution filter size was 3 × 3. The activation functions used in the networks were rectified linear unit (ReLU) [15] and sigmoid functions. The proposed network was optimized using an adaptive moment estimation (Adam) [16] optimizer with a learning rate of 0.001. The early stopping criteria were applied by monitoring the F-score with a patience value of 15.

Table 2: The hyperparameters and settings of the proposed network.

Parameter	Value
Convolution filter size	3 × 3
Activation function	ReLU, Sigmoid
Network initialzation	Glorot_uniform
Optimizer	Adam
Epochs	100
Learning rate	0.001
Early stopping method	Criteria = F-score Initial delay = 5 Patience = 15

3.3. Post-processing

After obtaining the frame based probabilities from the last layer, we applied two binary decision methods. First of all, it can be accomplished in a straightforward manner by setting frame decisions to 1, if the probability is over the threshold of 0.5. Alternatively, the result in each frame can be decided by using the Viterbi algorithm [17, 18]. Given a set of predictions that indicate the conditional probability of a condition, the Viterbi algorithm computes the most likely state sequence in the observations. Therefore, we obtained the binary result by applying the Viterbi algorithm to the frame probabilities in each class. Then, the results of binary decisions are segmented and smoothed in the time domain using the median filtering method. In this regard, the median filter size is a critical factor for the detection of the onset and offset of a sound event depending on its length. However, applying median filtering of the same length to various sound events is not recommended because each sound event has different characteristics. Therefore, we selected the median filter sizes according to the estimated lengths of the events. In this study, we selected various filter sizes for each class according to the median values of the predicted event lengths.

3.4. Experimental results

We evaluated the performance of the proposed approaches for weakly labeled semi-supervised SED. In these experiments, we explored our proposed methods: Inception CRNN, data augmentation (DA) for weakly labeled data, event activity detection (EAD) for strong label learning, single-length median filtering (SMF) with 51 frames, and multi-length median filtering (MMF). Table 3 shows the experimental results of the proposed Inception CRNN based SED. As shown in the table, the Inception CRNN shows about 4.5% better performance than the DCASE 2018 baseline system. In addition, as can be seen in the results, the DA method showed little SED performance improvement. Nevertheless, the DA method was used because of the improved sound tagging performance in the first stage. We also confirmed that the EAD for pseudo strong labeling could improve the SED performance by about 2.0%. Furthermore, by applying the Viterbi algorithm, a performance enhancement of 1.2% was obtained. Finally, about 6.5% improvement was achieved by applying the MMF method.

Table 3: Experimental results of the Inception CRNN based SED.

Method	F-score	Precision	Recall
DCASE 2018 Baseline + SMF51	14.06%	-	-
Inception CRNN + SMF51	18.5%	18.5%	20.0%
Inception CRNN + DA + SMF51	18.9%	19.6%	19.6%
Inception CRNN + DA + EAD + SMF51 (Submission-1)	21.9%	23.3%	23.3%
Inception CRNN + DA + EAD + Viterbi + SMF51 (Submission-2)	23.1%	26.0%	23.4%
Inception CRNN + DA + EAD + MMF (Submission-3)	28.4%	28.3%	31.0%
Inception CRNN + DA + EAD + Viterbi + MMF (Submission-4)	29.3%	30.1%	30.3%

Consequently, the system that used MMF with the Viterbi algorithm showed the best performance. Compared to the DCASE 2018 baseline system, the proposed model achieved an approximately 15.24% gain in the F-score. Table 4 shows detailed experimental results of the proposed Inception CRNN with DA, EAD, Viterbi, and MMF.

Table 4: Detailed experimental results of the proposed system. (Submission-4)

Event label	F-score	Precision	Recall
Alarm/bell/ringing	28.4%	28.3%	28.6%
Blender	28.9%	26.0%	32.5%
Cat	12.6%	17.8%	9.8%
Dishes	10.6%	10.6%	10.7%
Dog	26.7%	30.3%	23.8%
Electric shaver/toothbrush	48.1%	50.0%	46.4%
Frying	13.3%	9.1%	25.0%
Running water	34.9%	28.6%	44.7%
Speech	16.5%	15.2%	18.1%
Vacuum cleaner	73.0%	85.2%	63.9%

4. CONCLUSION

In this paper, the Inception CRNN method that observes the various receptive fields in each CNN layer is proposed for large-scale detection of sound events using small weakly labeled data. By applying the proposed network structure to the SED system, it was shown that the Inception CRNN model achieves a better result than the baseline model. We also proposed various performance enhancement methods such as DA, EAD, Viterbi algorithm, and MMF to improve the SED performance of the proposed system. In conclusion, it was confirmed that the proposed method provide a higher SED result than the baseline system.

5. ACKNOWLEDGMENT

This work was supported by Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea government (MSIT) (No.2017-0-00050, Development of Human Enhancement Technology for auditory and muscle support)

6. REFERENCES

[1] D. Stowell, D. Giannoulis, E. Benetos, M. Lagrange, and M.D. Plumbley. “Detection and classification of acoustic scenes and events,” IEEE Transactions on Multimedia, 17(10):1733–1746, Oct 2015.

[2] A. Mesaros, T. Heittola, E. Benetos, P. Foster, M. Lagrange, T. Virtanen, and M. D. Plumbley. “Detection and classification of acoustic scenes and events: outcome of the dcase 2016 challenge,” IEEE/ACM Transactions on Audio, Speech, and Language Processing, 26(2): 379–393, Feb 2018.

[3] http://www.cs.tut.fi/sgn/arg/dcase2017/.

[4] http://dcase.community/challenge2018/.

[5] Y. Xu, Q. Kong, W. Wang, and M. D. Plumbley, “Surrey-cvssp system for DCASE2017 challenge task4,” in Proceedings of the Detection and Classification of Acoustic Scenes and Events Workshop (DCASE), 2017.

[6] D. Lee, S. Lee, Y. Han, and K. Lee, “Ensemble of Convolutional Neural Networks for Weakly-Supervised Sound Event Detection Using Multiple Scale Input,” in Proceedings of the Detection and Classification of Acoustic Scenes and Events Workshop (DCASE), 2017.

[7] J. Lee, J. Park, S. Kum, Y. Jeong, and J. Nam, “Combining Multi-Scale Features Using Sample-Level Deep Convolutional Neural Networks for Weakly Supervised Sound Event Detection,” in Proceedings of the Detection and Classification of Acoustic Scenes and Events Workshop (DCASE), 2017.

[8] S. Adavanne, and T. Virtanen, “Sound Event Detection Using Weakly Labeled Dataset with Stacked Convolutional and Recurrent Neural Network,” in Proceedings of the Detection and Classification of Acoustic Scenes and Events Workshop (DCASE), 2017.

[9] J. Salamon, B. McFee, and P. Li, “DCASE 2017 Submission: Multiple Instance Learning for Sound Event Detection,” in Proceedings of the Detection and Classification of Acoustic Scenes and Events Workshop (DCASE), 2017.

[10] R. Serizel, N. Turpault, H. Eghbal-Zadeh, and A. Parag Shah, “Large-Scale Weakly Labeled Semi-Supervised Sound Event Detection in Domestic Environments,” in Proceedings of the Detection and Classification of Acoustic Scenes and Events Workshop (DCASE), 2018.

[11] B. McFee, E. Humphrey, and J. Bello, “A software framework for musical data augmentation,” in Proceedings of the 19th International Society for Music Information Retrieval Conference (ISMIR), 2015.

[12] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, “Going deeper with convolutions,” in Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR), 2015.

[13] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, Z. Wojna, “Rethinking the Inception architecture for computer vision,” in Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR), 2016.

[14] C. Szegedy, S. Ioffe, V. Vanhoucke, A. Alemi, “Inception-v4, Inception-Resnet and the impact of residual connections on learning,” in Proceedings of the AAAI Conference on Artificial Intelligence (AAAI), 2017.

[15] V. Nair and G. E. Hinton, “Rectified linear units improve restricted Boltzmann machines,” in Proceedings of the 27th International Conference on Machine Learning (ICML), 2010.

[16] D. P. Kingma and J. L. Ba, “Adam: A method for stochastic optimization,” in Proceedings of the 3rd International Conference on Learning Representations (ICLR), 2015.

[17] Viterbi, Andrew. “Error bounds for convolutional codes and an asymptotically optimum decoding algorithm,” IEEE transactions on Information Theory, 13(2): 260-269, 1967.

[18] N. Ryant, M. Libeman, J. Yuan, “Speech activity detection on YouTube using deep neural network,” in Proceeding of 14th Annual Conference of the International Speech Communication Association (INTERSPEECH), August 25-29, Lyon, France, pp. 728-731, 2013.

Polyphonic audio tagging with sequentially labelled data using CRNN with learnable gated linear units

Yuanbo Hou[1], *Qiuqiang Kong*[2], *Jun Wang*[1], *Shengchen Li*[1]

[1] Beijing University of Posts and Telecommunications, Beijing, P.R.China
{hyb, shengchen.li, wangjun19930314}@bupt.edu.cn
[2] Centre for Vision, Speech and Signal Processing, University of Surrey, UK
q.kong@surrey.ac.uk

ABSTRACT

Audio tagging aims to detect the types of sound events occurring in an audio recording. To tag the polyphonic audio recordings, we propose to use Connectionist Temporal Classification (CTC) loss function on the top of Convolutional Recurrent Neural Network (CRNN) with learnable Gated Linear Units (GLU-CTC), based on a new type of audio label data: Sequentially Labelled Data (SLD). In GLU-CTC, CTC objective function maps the frame-level probability of labels to clip-level probability of labels. To compare the mapping ability of GLU-CTC for sound events, we train a CRNN with GLU based on Global Max Pooling (GLU-GMP) and a CRNN with GLU based on Global Average Pooling (GLU-GAP). And we also compare the proposed GLU-CTC system with the baseline system, which is a CRNN trained using CTC loss function without GLU. The experiments show that the GLU-CTC achieves an Area Under Curve (AUC) score of 0.882 in audio tagging, outperforming the GLU-GMP of 0.803, GLU-GAP of 0.766 and baseline system of 0.837. That means based on the same CRNN model with GLU, the performance of CTC mapping is better than the GMP and GAP mapping. Given both based on the CTC mapping, the CRNN with GLU outperforms the CRNN without GLU.

Index Terms— Audio tagging, Convolutional Recurrent Neural Network (CRNN), Gated Linear Units (GLU), Connectionist Temporal Classification (CTC), Sequentially Labelled Data (SLD)

1. INTRODUCTION

Audio tagging aims to detect the types of sound events occurring in an audio recording. Audio recordings are typically short segments such as the audio recordings in IEEE AASP DCASE 2018 Challenge Task 4 [1]. Audio tagging has many applications in information retrieval [2], audio classification [3], acoustic scene recognition [4] and industry sound recognition [5].

Most previous works of audio tagging relies on strongly labelled data or weakly labelled data. In strongly labelled data [4], each audio clip is labelled with both the tags and the onset and offset times of sound events. However, labelling strong label is time consuming and labor expensive, resulting strongly labelled data is scarce and its size is often limited to minutes or a few hours [6]. Thus the audio research community have turned to large-scale datasets without the onset and offset times of sound events, which is referred to as Weakly Labelled Data (WLD) [7]. WLD is also called clip level labelled data. In WLD, only the presence or absence of sound events are known, but the occurrence sequence of sound events are not known.

In this paper, we explore the possibility of Sequentially Labelled Data (SLD) in real-life polyphonic audio tagging. SLD is a type of audio label newly proposed in [8]. In SLD, both the tags of audio clip and the sequence of tags are known, without the onset and the offset of tags. SLD reduces the workload of data annotation and avoids the problem of inaccurate onset and offset annotation of tags in strongly labelled data. In addition, SLD contains the sequential information of tags which is not provided by WLD [8]. However, in the previous work [8], the SLD was the synthesized monophonic audio based on IEEE DCASE 2013 development dataset, there is no overlap between sound events. To explore the possibility of SLD in real-life audio recordings, we manually label 1578 polyphonic audios of DCASE 2018 Task 4 with sequential labels and release it here[1]. The details of sequential labelling of polyphonic audio recordings will be introduced in Section 3.

To predict the sequential labels of SLD in polyphonic audio recordings, we propose to use CTC loss function on the top of CRNN with learnable Gated Linear Units (GLU-CTC). This idea is inspired by the great performance of CTC in Automatic Speech Recognition [9]. CTC is a learning technique for sequence labelling with RNN, which allows RNN to be trained for sequence-to-sequence tasks without requiring any prior alignment between the input and target sequences. In GLU-CTC, CTC objective function maps the frame-level probability of sound events to the target sequential labels of sound events, similar to the pooling layer in neural networks. So we explore the performance of this three pooling function: CTC, Global Max Pooling (GMP) and Global Average Pooling (GAP) in polyphonic audio tagging, based on the same CRNN with GLU. This three models are abbreviated as GLU-CTC, GLU-GMP and GLU-GAP, respectively. In this paper, the baseline system is a CRNN without GLU train with CTC loss function.

There are two contributions in this paper. First, in polyphonic audio tagging we explore the possibility of a new label type: Sequentially Labelled Data, which not only reduces the workload of data annotation in strong labels, but also indicates the sequential information of tags in weak labels. We release the SLD of DCASE 2018 Task 4 in here[1]. Second, to predict the sequential labels of SLD in polyphonic audio recordings, we

[1] https://github.com/moses1994/DCASE2018-Task4

propose to use CTC learning technique to train a CRNN model with learnable GLU. And we compare the performance of GLU-CTC, GLU-GMP, GLU-GAP and the baseline system, which is a CRNN train with CTC loss function. There is no GLU in baseline system.

This paper is organized as follows, Section 2 introduces related works. Section 3 describes the annotation method of SLD in polyphonic audio recordings. Section 4 describes how the CTC uses SLD in polyphonic audio tagging and the model structure. Section 5 describes the dataset, experimental setup and results. Section 6 gives conclusions.

2. RELATED WORK

Audio classification and detection have obtained increasing attention in recent years. There are many challenges for audio detection and tagging such as IEEE AASP challenge on DCASE 2013 [4], DCASE 2016 [10] and DCASE 2017 [6].

Many conventional works of audio classification and audio clip tagging used Mel Frequency Cepstrum Coefficient (MFCC) and Gaussian Mixture Model (GMM) as baseline system [4]. Recent audio classification methods including Deep Neural Networks (DNNs) [6], Convolution Neural Networks (CNNs) [11] and Recurrent Neural Networks (RNN) [3], with inputs varying from Short-Time Fourier Transform (STFT), Mel energy, spectrogram, MFCC to Constant Q Transform (CQT) [12].

The bag of frames (BOF) model was used in [13], where an audio clip is cut into segments and each segment inherits the labels of the audio clip. BOF is based on an assumption that tags occur in all frames, which is however not the case in practice. Some sound events such as "gunshot" only happen a short time in an audio clip. State-of-the-art audio tagging methods [14] transform waveform to the Time-Frequency (T-F) representation. Then, the T-F representation is treated as an image which is fed into CNNs. However, unlike image where the objects usually occupy a dominant part of an image, in an audio clip events only occur a short time. To solve this problem, attention models [15] for audio tagging and classification are applied to attend to the audio events and ignore the back ground sounds.

3. SEQUENTIALLY LABELLED DATA

The polyphonic audio data used in this paper is the weak annotations training set of DCASE 2018 Task 4, a subset of Google Audio Set [16]. Audio Set consists of an ontology of 632 sound event classes and a collection of 2 million human-labeled 10-second audio clips drawn from YouTube [16].

In the training set, the polyphony makes it hard to define ordered sequences of sound events. To tackle this problem, we use the order of boundaries of each sound event, the order of onset and offset, but not the time stamps as the sequential labels. For example, we could use the sequential labels *dishes_start, dishes_end, dishes_start, dishes_end, speech_start, blender_start, speech_end, speech_start, blend_end, speech_end* as the sequential label for the audio clip in Fig. 1. Another example is if the content of an audio clip could be described by *a dog barks while a car rings*, we used the sequential labels *ring_start, dog_start, dog_end, ring_end* as the sequential label. In the ground truth label sequence, the tags of the audio clip and the

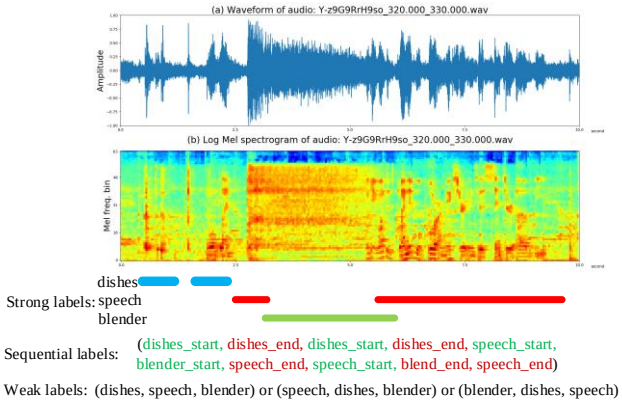


Figure 1: From top to bottom: (a) waveform of an audio clip containing three sound events: "dishes, speech, blender"; (b) log Mel spectrogram of (a); Strong labels, sequential labels and weak labels of the audio clip.

sequence of tags are known, without knowing their occurrence time. We refer to the audio clip labelled by label sequence as Sequentially Labelled Data (SLD). Fig. 1 shows an audio clip with strong, sequential and weak tags.

In this paper, we manually labelled the weak annotations training set of DCASE 2018 Task 4 with sequential labels and release it after verification. See here[1] for more details about SLD.

4. METHOD

In this section, we will explain how to use CTC in polyphonic audio tagging based on SLD. Then, we will describe the model structure used in this paper.

4.1. CTC in Polyphonic Audio Tagging using SLD

CTC is a learning technique for sequence labelling, it shows a new way for training RNN with unsegment sequences. In fact, CTC redefines the loss function of RNN [17] and allows RNN to be trained for sequence-to-sequence tasks, without requiring any prior alignment (*i.e.* starting and ending time of sound events) between the input and target sequences [9]. In audio tagging, we are only interested in the label sequence of corresponding audio clip, not the ground truth alignment of events in the audio clip. Thus, we want to marginalize out the alignment.

To marginalize out the alignment, first, CTC adds an extra "blank" label (denoted by "-") to original label set L [9]. Then, it defines a many-to-one mapping β that transforms the alignment (*i.e.* the sequence of output labels at each time step, also called a

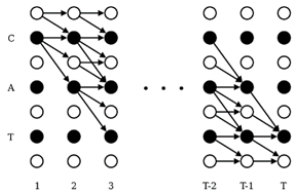


Figure 2: Trellis for computing CTC loss function [17] applied to labelling 'CAT'. Black circles represent labels, white circles represent blanks. Arrows signify allowed transitions.

path [17]) to label sequence. The mapping β removes repeated labels from the path to a single one, then removes the "blank" labels. For example, β(C-AT-)=β(-CC--ATT)=CAT, that is, path 'C-AT-' and '-CC--ATT' both map to the label sequence 'CAT'.

The CTC objective function is defined as the negative logarithm of the total probability of all paths [9] that map to the ground truth label sequence. The total probability can be found using dynamic programming algorithm [17] on the trellis shown in Fig. 2. On the x-axis is time steps, on the y-axis is "modified label sequence", that is target label sequence with blank labels added to the beginning and the end and inserted between every pair of labels.

When we use the simple best path decoding to decode the output of CTC, the output of CTC is directly the label sequence. By this means no threshold is needed to determine whether there are corresponding events in the audio clip. This will reduce the risk of over-fitting due to specific thresholds, which is an advantage of using CTC loss function in audio tagging. More details about CTC can be seen [17].

4.2. Model Structure

Inspired by the good performance of CRNN in audio tagging [15], CRNN is used in this paper shown in Fig. 3. First, the waveforms of audio clips are transformed to T-F representations such as Mel spectrograms. And convolutional layers are applied on the T-F representations to extract high level features. Next, Bidirectional Gated Recurrent Units (BGRU) are adopted to capture the temporal context information. Finally, the output layer is a dense layer with sigmoid activation function since audio tagging is a multi-class classification problem [3, 6].

In the CRNN, the output activation from the CNN layers are padded with zeros to keep the dimension of the output the same as input. And the max-pooling is applied in the frequency

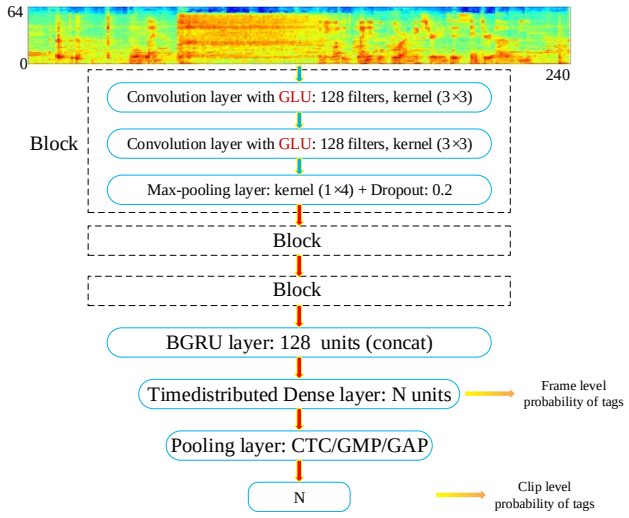


Figure 3: Model Structure. Due to the acoustic event classes number is 10 in DCASE 2018 Task 4, thus, for model with GMP and GAP layer, N=10. For model with CTC layer, N=21 (10 *2+1), the extra '1' indicates the blank label.

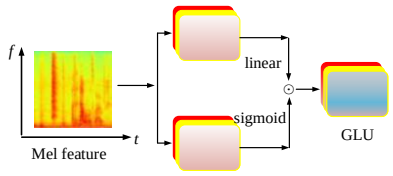


Figure 4: The Structure of GLU.

axis only to preserve the time resolution of the input. Clip level probability of tags can be obtained from the last layer. To compare the performance of different pooling function, there are three pooling operations in Fig. 3, CTC, Global Max Pooling (GMP) and Global Average Pooling (GAP).

4.3. Gated Linear Units

As shown in Fig. 3, a CRNN model with 13 layers is applied for audio tagging. In order to reduce the gradient vanishing problem in deep networks, the Gated Linear Units (GLU) [18] is used as the activation function to replace the ReLU [19] activation function in the CRNN model. The structure of GLU is shown in Fig. 4. By providing a linear path for the gradients propagation while keeping nonlinear capabilities through the sigmoid operation, GLU can reduce the gradient vanishing problem for deep networks [18]. Similar to the gating mechanisms in long short-term memories [20] or gated recurrent units [21], GLU can control the amount of information of a T-F unit flow to the next layer. GLU are defined as:

$$Y = (W * X + b) \odot \sigma(V * X + c) \quad (1)$$

where σ is sigmoid function, the symbol $\odot$ is the element-wise product and $*$ is the convolution operator. W and V are convolutional filters, b and c are biases. X denotes the input T-F representation in the first layer or the feature maps of the interval layers in model.

The value of sigmoid function ranges from 0 to 1, so if a GLU gate value is close to 1, then the corresponding T-F unit is attended. If a GLU gate value is near to 0, then the corresponding T-F unit is ignored. By this means the network can learn to attend to sound events and ignore the unrelated sounds.

5. EXPERIMENTS AND RESULTS

5.1. Dataset, Experiments Setup and Evaluation Metrics

In this paper, the training set is 1578 clips (2244 class occurrences) of Task 4 from domestic environments, which consists of 10 classes of sound events. We manually label the 1578 audio clips with sequential labels and release it after verification, the annotation method is described in Section 3. The test set is 288 polyphonic audio clips (906 events) of Task 4 [1].

For all the models described in this paper, in training, log Mel band energy is extracted in Hamming window of length 64 ms with 64 Mel frequency bins [22]. For a given audio clip of 10-second in Task 4, this feature extraction block results in a (240×64) output as shown in Fig. 3. 240 is the number of frames

Table 1: Averaged Stats of Audio Tagging

Metric	*AUC of each event class*										*avg.*			
Event	*Speech*	*Dog*	*Cat*	*Bell*	*Dishes*	*Frying*	*Blender*	*Water*	*cleaner*	*Shaver*	*AUC*	*Precision*	*Recall*	*F-score*
GLU-GAP	0.895	0.946	0.875	0.820	0.583	0.602	0.641	0.773	0.771	0.758	0.766	**0.960**	0.588	0.730
GLU-GMP	0.909	0.946	0.921	0.873	0.669	0.643	0.691	0.813	0.785	0.778	0.803	0.957	0.645	0.771
GLU-CTC	**0.941**	**0.969**	**0.994**	**0.942**	**0.762**	**0.905**	0.753	**0.860**	**0.850**	**0.835**	**0.882**	0.816	**0.816**	**0.816**
Baseline	0.912	0.953	0.957	0.836	0.684	0.776	**0.795**	0.839	0.808	0.808	0.837	0.706	0.763	0.734

and 64 is the number of Mel frequency bins. The binary cross-entropy loss [23] is applied between the predicted probability of each tag and the corresponding ground truth tag. Dropout and early stopping criteria are used in training phase to prevent over-fitting. The model is trained for maximum 200 epochs with Adam optimizer with a learning rate of 0.001.

To evaluate the results of audio tagging in clip level in this paper, we follow the metrics proposed in [22]. The results are evaluated by *Precision* (*P*), *Recall* (*R*) and *F-score* [24] and Area Under Curve (*AUC*) [25]. Larger *P*, *R*, *F-score* and *AUC* indicates better performance.

5.2. Results

In this paper, the GLU-CTC, GLU-GMP and GLU-GAP all contain the learnable GLU, which introduces the attention mechanism in the convolutional layers in CRNN. However, there is no GLU in the baseline model, which is a CRNN train with CTC objective function. To evaluate the performance of the models in this paper, we calculate the *AUC* score of audio tagging results in clip level of these models. As shown in Table 1, GLU-CTC achieves an averaged *AUC* of 0.882 outperforming the GLU-GAP and GLU-GMP, and also better than the baseline system. Table 1 also shows the averaged statistic including *Precision*, *Recall*, *F-score* and *AUC* over 10 kinds of sound events, respectively. GLU-CTC mapping performs better than GLU-GAP and GLU-GMP, too. That is, based on the same CRNN model with GLU, the performance of CTC mapping function is better than the GAP and GMP mapping function in polyphonic audio tagging.

The averaged stats of audio tagging is evaluated in clip level of audio clips, the frame level predictions of models on example audio clip was shown in Fig. 5. In Fig. 5, the predictions of GLU-GAP in frame level is always 1, which means the predictions of GLU-GAP in frame level overestimates the occurrence probability of corresponding event. While GLU-GMP, in contrast, underestimates it. GLU-GMP produces wide peaks, indicating the onset and offset times of event. That shows max pooling has ability to locate event, while average pooling seems to fail. The reason may be max pooling encourages the response for a single location to be high [26], for similar audio events which can obtain similar features. While average pooling encourages all response to be high [26], difference features of each event make it difficult to locate event.

In Fig. 5, the GLU-CTC could predict the onset (start) and offset (end) tag sequence of corresponding audio recording, typically as a series of spikes [17]. Although the spikes align well with the actual position of the boundaries of sound events in audio recording, there is no time span information about these events. The spikes outputted by GLU-CTC could locate corresponding events in the audio clip, while baseline system seems to fail, which means the attention mechanism introduced by GLU is helpful for audio tagging. The reason may be the attention introduced by GLU focuses on the local information within each feature map, which could help GLU-CTC better learn the high-level representations of corresponding audio events.

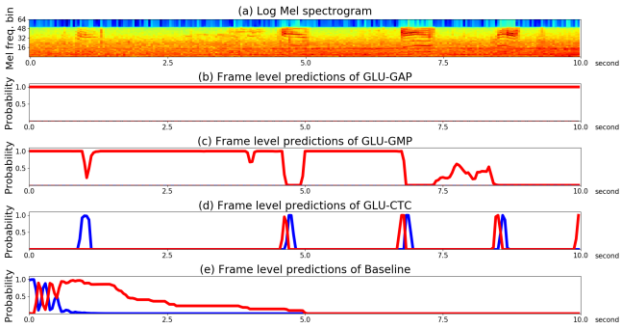


Figure 5: Frame level predictions of GLU-GAP (b), GLU-GMP (c), GLU-CTC (d), and Baseline (e). In GLU-CTC and Baseline, blue peaks denote the starting and red peaks denote the ending of corresponding sound events.

6. CONCLUSION

In this paper, we explore the possibility of a new type of audio label data called SLD in polyphonic audio tagging. To utilize SLD in audio tagging, we propose a GLU-CTC model. In GLU-CTC, CTC layer maps frame level tags to clip level tags, similar to the pooling operations. Experiments show GLU-CTC outperforms GLU-GAP and GLU-GMP. Finally, we released the sequential labels of DCASE 2018 Task 4 after verification. In the future, we will explore the possibility of SLD in sound event detection with polyphonic audio recordings and try to expand the size of SLD.

7. ACKNOWLEDGMENT

Qiuqiang Kong was supported by EPSRC grant EP/N014111/1 ``Making Sense of Sounds" and a Research Scholarship from the China Scholarship Council (CSC) No. 201406150082."

8. REFERENCES

[1] http://dcase.community/challenge2018/task-large-scale-weakly-labeled-semi-supervised-sound-event-detection.

[2] G. Guo and Stan Z Li, "Content-based audio classification and retrieval by support vector machines," *IEEE Transactions on Neural Networks*, vol. 14, no. 1, pp. 209–215, 2003

[3] Y. Xu, Q. Kong and W. Wang, et al. "Large-scale weakly supervised audio classification using gated convolutional neural network," arXiv preprint arXiv: 1710.00343, 2017.

[4] D. Stowell, D. Giannoulis and E. Benetos, et al. "Detection and classification of acoustic scenes and events," *IEEE Transactions on Multimedia*, vol. 17, no. 10, pp. 1733–1746, 2015

[5] S. Dimitrov, J. Britz, B. Brandherm, and J. Frey, "Analyzing sounds of home environment for device recognition," in *AmI. Springer*, 2014, pp. 1–16.

[6] A. Mesaros, T. Heittola, A. Diment and B. Elizalde, et al. "DCASE 2017 challenge setup: Tasks, datasets and baseline system," in *Proceedings of DCASE2017 Workshop*.

[7] A. Kumar and B. Raj, "Audio event detection using weakly labelled data," in *Proceedings of the 2016 ACM on Multimedia Conference*. *ACM*, 2016, pp. 1038–1047

[8] https://www.researchgate.net/publication/326588286_Audio_Tagging_With_Connectionist_Temporal_Classification_Model_Using_Sequentially_Labelled_Data

[9] A. Graves and N. Jaitly, "Towards end-to-end speech recognition with recurrent neural networks", in *Proc. of ICML*, 2014.

[10] M. Valenti, A. Diment and G. Parascandolo, et al., "DCASE 2016 acoustic scene classification using convolutional neural networks," *Workshop on Detection and Classification of Acoustic Scenes and Events (DCASE 2016)*, Budapest, Hungary, 2016

[11] Y. Han and K. Lee, "Acoustic scene classification using convolutional neural network and multiple-width frequency-delta data augmentation," arXiv preprint arXiv: 1607.02383, 2016.

[12] T. Lidy and A. Schindler, "CQT-based convolutional neural networks for audio scene classification," in *Workshop on Detection and Classification of Acoustic Scenes and Events (DCASE 2016)*, Budapest, Hungary, 2016

[13] J. Ye, T. Kobayashi, M. Murakawa, and T. Higuchi, "Acoustic scene classification based on sound textures and events," in *Proceedings of ACM on Multimedia Conference. ACM*, 2015, pp. 1291–1294.

[14] K. Choi, G. Fazekas, and M. Sandler, "Automatic tagging using deep convolutional neural networks," arXiv preprint arXiv: 1606.00298, 2016.

[15] Y. Xu, Q. Kong, Q. Huang, W. Wang, and M. D. Plumbley, "Attention and localization based on a deep convolutional recurrent model for weakly supervised audio tagging," in *INTERSPEECH. IEEE*, 2017, pp. 3083–3087.

[16] Gemmeke, Jort F., et al. "Audio Set: An ontology and human-labeled dataset for audio events." *IEEE International Conference on Acoustics, Speech and Signal Processing IEEE*, 2017:776-780.

[17] Graves A, Gomez F. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks[C]. *International Conference on Machine Learning. ACM*, 2006:369-376.

[18] Y. N. Dauphin, A. Fan, M. Auli, and D. Grangier, "Language modelling with gated convolutional networks," arXiv preprint arXiv: 1612.08083, 2016.

[19] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in ICML, 2010, pp. 807–814.

[20] S. Hochreiter and J. Schmidhuber, "Long short-term memory," Neural Computation, vol. 9, no. 8, pp. 1735–1780, 1997.

[21] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," arXiv preprint arXiv: 1412.3555, 2014.

[22] Kong Q, Xu Y, Sobieraj I, et al. Sound Event Detection and Time-Frequency Segmentation from Weakly Labelled Data, arXiv preprint arXiv: 1804.04715, 2018.

[23] F. Ghazani, and M. S. Baghshah. "Multi-label classification with feature-aware implicit encoding and generalized cross-entropy loss." *Electrical Engineering IEEE*, 2016:1574-1579.

[24] A. Mesaros, T. Heittola, and T. Virtanen, "Metrics for polyphonic sound event detection," *Applied Sciences*, vol. 6, no. 6, p. 162, 2016.

[25] J. A. Hanley and B. J. McNeil, "The meaning and use of the area under a receiver operating characteristic (roc) curve." *Radiology*, vol. 143, no. 1, pp. 29–36, 1982.

[26] Kolesnikov, Alexander, and C. H. Lampert. "Seed, Expand and Constrain: Three Principles for Weakly-Supervised Image Segmentation." *European Conference on Computer Vision Springer International Publishing*, 2016:695-711.

SOUND EVENT DETECTION USING WEAKLY LABELLED SEMI-SUPERVISED DATA WITH GCRNNS, VAT AND SELF-ADAPTIVE LABEL REFINEMENT

Robert Harb

Graz University of Technology, Austria
robert.harb@student.tugraz.at

Franz Pernkopf

Graz University of Technology, Austria
Signal Processing and Speech Communication Laboratory
pernkopf@tugraz.at

ABSTRACT

In this paper, we present a gated convolutional recurrent neural network based approach to solve task 4, large-scale weakly labelled semi-supervised sound event detection in domestic environments, of the DCASE 2018 challenge. Gated linear units and a temporal attention layer are used to predict the onset and offset of sound events in 10s long audio clips. Whereby for training only weakly-labelled data is used. Virtual adversarial training is used for regularization, utilizing both labelled and unlabelled data. Furthermore, we introduce self-adaptive label refinement, a method which allows unsupervised adaption of our trained system to refine the accuracy of frame-level class predictions. The proposed system reaches an overall macro averaged event-based F-score of 34.6%, resulting in a relative improvement of 20.5% over the baseline system.

***Index Terms*—** DCASE 2018, Convolutional neural networks, Sound event detection, Weakly-supervised learning, Semi-supervised learning

1. INTRODUCTION

In this paper we summarize the methods we use to solve task 4 [1] of the DCASE 2018 challenge, the *large-scale weakly labelled semi-supervised sound event detection in domestic environments*. In contrast to audio tagging (AT), sound event detection (SED) not only requires to detect the presence of an event, but also a prediction about the temporal location in a given audio recording. Whereby in the data provided by the DCASE challenge, one input sequence possibly contains multiple occurrences of different event classes with potential temporal overlaps. Additionally, the training data is only weakly labelled. Therefore for training, the labels of each clip contain only information about the presence or absence of an event, but no strong labels which indicate the exact temporal onset and offset.

The proposed method uses a gated convolutional recurrent neural network (GCRNN). This is similar to the best model of last years DCASE 2017 challenge task 4 [2] which also used a GCRNN based approach. Although, the objective of the 2017 and 2018 DCASE challenge is SED, there are significant differences in the structure of the provided training data and evaluation metric. More precisely, the following changes have been made at the 2018 challenge:

- The amount of weakly labelled training data is significantly smaller, 1,578 compared to 51,172.
- In addition to the weakly labelled training set, there are unlabelled in-domain and unlabelled out-of-domain sets provided.
- The domain of the events is different: *domestic environments* compared to *smart cars*. Whereby the number of classes decreased from 17 to 10.
- For evaluation, an event-based F-score with a 200ms collar on onsets and offsets is used, instead of a segment-based error rate which is determined of one-second segments.

With our work we show that a GCRNN based approach for SED similar to [2], is also suitable in a setting with the aforementioned differences. Whereby we introduce two major changes:

First, to incorporate the provided unlabelled data we use virtual adversarial training (VAT) [3]. VAT has, amongst others, already been used successfully in semi-supervised text [4], image classification [3], acoustic event detection [5] and phone classification [6] tasks. Furthermore, VAT showed competitive performance against other deep semi-supervised learning algorithms [7].

Secondly, as an extension to the attention mechanism we introduce an algorithm we call self-adaptive label refinement (SALR), which uses unlabelled input data and clip-level class predictions to refine the frame-level predictions of our model.

2. PROPOSED METHOD

2.1. Gated convolutional recurrent neural network

The winning team of last year's DCASE SED task [2] showed that using gated linear units (GLUs) [8] instead of commonly used activation functions like rectified linear units (RELUs) or leaky ReLUs in the CRNN is a useful approach for SED.

Gating mechanisms have been used successfully in a variety of neural network architectures. For example in RNNs using LSTM [9] cells, which have a separate input, output and forget gate. The rough idea behind gating mechanisms is to have a gate which can control how information flows in the network.

In the setting of SED, the GLU units should adapt their behaviour such that they act as an attention mechanism on the time-frequency (T-F) bin of each feature map. They can set their value close to one if information related to any of the considered audio events passes through, and otherwise block the flow of unrelated information by setting their value close to zero.
GLUs are defined as follows:

$$\mathbf{Y} = (\mathbf{W} * \mathbf{X} + \mathbf{b}) \odot \sigma(\mathbf{V} * \mathbf{X} + \mathbf{c}), \tag{1}$$

where $\mathbf{W}$ and $\mathbf{V}$ denote the convolutional filters with their respective biases $\mathbf{b}$ and $\mathbf{c}$, σ is the sigmoid function, $\mathbf{X}$ denotes the input to the layer, and $\odot$ denotes elementwise multiplication.

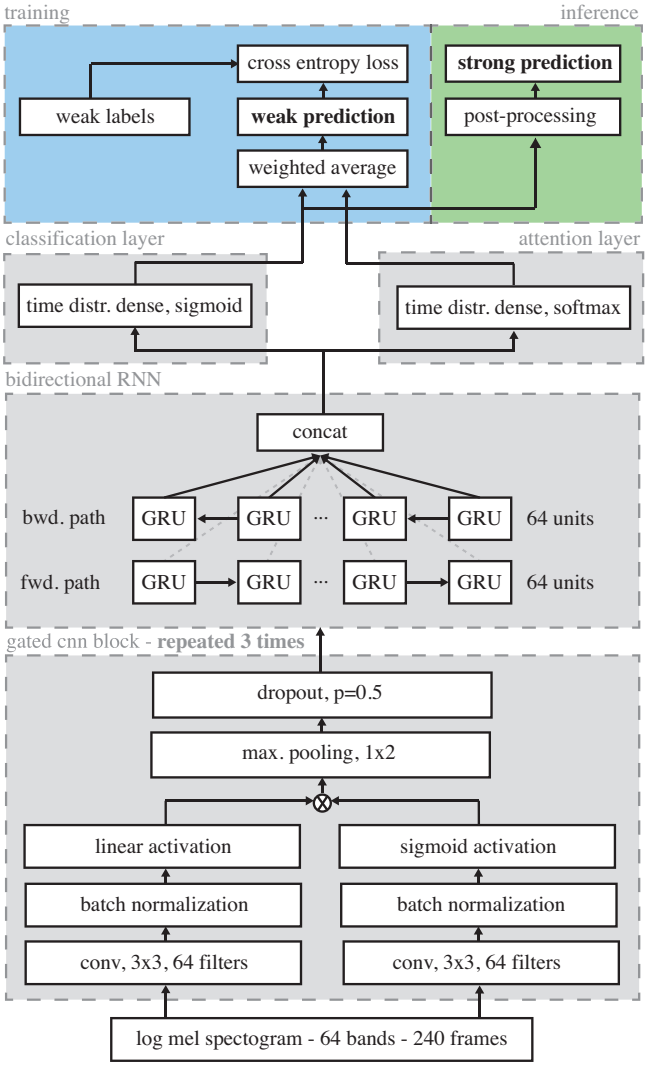


Figure 1: Network structure

Figure 1 shows how the gated CNN blocks are incorporated into the network, whereby in our model we use three subsequent gated CNN blocks.

The gated CNN blocks are followed by a bidirectional RNN containing 64 units in the forward and backward path, their output is concatenated and passed to the attention and classification layer which are described in Section 2.3.

The final prediction y_c for the weak label of class c is determined by the weighted average of the element-wise multiplication of the attention and classification layer output of class c:

$$y_c = \frac{\sum_t^T \mathbf{z}_c^{cla}(t) \odot \mathbf{z}_c^{att}(t)}{\sum_t^T \mathbf{z}_c^{att}(t)}, \tag{2}$$

where $\mathbf{z}_c^{cla}$(t) and $\mathbf{z}_c^{att}$(t) are the outputs of the classification layer and of the attention layer of class c. T denotes the frame-level resolution of the input spectrogram, which is equal to the resolution of $\mathbf{z}_c^{cla}$(t) and $\mathbf{z}_c^{att}$(t), and t is the frame index.

2.2. Virtual adversarial training

We make use of VAT [3] for regularization. We calculate the virtual adversarial loss such that the robustness of the model's posterior distribution of predictions at clip-level $p(\mathbf{y}|\mathbf{x})$ is increased for small and bounded perturbations of the log-scaled mel-spectrograms $\mathbf{x}$.

The adversarial perturbation $\mathbf{r}_{v\text{-}adv}$ is computed by maximizing a non-negative distance function between the unperturbed $p(\mathbf{y}|\mathbf{x};\theta)$ and perturbed $p(\mathbf{y}|\mathbf{x}+\mathbf{r};\theta)$ posterior. Whereby θ denotes the current model parameter. The Kullback-Leibler divergence KL is used as distance function between $p(\mathbf{y}|\mathbf{x};\theta)$ and $p(\mathbf{y}|\mathbf{x}+\mathbf{r};\theta)$, and $||\mathbf{r}||$ is limited to the sphere around $\mathbf{x}$ with some radius $\leq \epsilon$, i.e. $\mathbf{r}_{v\text{-}adv}$ is determined as

$$\mathbf{r}_{v\text{-}adv} = \arg \max_{\mathbf{r},\|\mathbf{r}\|\leq\epsilon} \mathrm{KL}[p(\mathbf{y}|\mathbf{x};\theta)||p(\mathbf{y}|\mathbf{x}+\mathbf{r};\theta)]. \tag{3}$$

There is no evident closed-form solution for $\mathbf{r}_{v\text{-}adv}$, but [3] gives a detailed derivation how to calculate an approximation of $\mathbf{r}_{v\text{-}adv}$. When using VAT the following additional cost is added to the objective function:

$$\mathrm{KL}[p(\mathbf{y}|\mathbf{x};\theta)||p(\mathbf{y}|\mathbf{x}+\mathbf{r}_{v\text{-}adv};\theta)]. \tag{4}$$

Since calculating the virtual adversarial perturbation only requires input $\mathbf{x}$ and does not require label $\mathbf{y}$, VAT is applicable to semi-supervised training. Therefore we use it to incorporate the unlabelled in-domain dataset into training. However, we decided not to include any of the provided unlabelled out-of-domain data since it has been shown previously that adding unlabelled data from different classes than the labelled data, can actually decrease the performance of semi-supervised learning algorithms like VAT [7].

2.3. Attention mechanism

To predict the temporal locations of each audio event which is presented in a given input sample, we use a similar approach as used in [2]. We extend it by introducing self-adaptive label refinement based on weak and strong prediction alignment. This selects for each event class an appropriate post-processing on the networks attention output. In the following the term weak prediction is used to refer to predictions at clip-level and strong prediction is used to refer to class predictions at frame-level.

As depicted in Figure 1, the output of a bidirectional RNN is fed into both an attention and a classification layer. The classification layer uses a sigmoid activation function to predict the probability of each occurring class at each timestep. While the attention layer uses a softmax activation over all classes. Intuitively, using a softmax in the attention layer should aid the network to learn to pick the most dominant class at each frame. Although this might not be an ideal approach if temporal overlaps of multiple events are occurring, since then a more dominant event might be able to suppress the activation of another one.

Figure 2 shows the output of the classification and attention layer for one audio clip of the development set containing several events labelled as dog. It can be seen that there is a clear correlation between ground truth event labels and the activations of the attention and classification layer. However it is not obvious how to extract the exact start and end points of each individual event from the layer activations. Our experiments showed that just taking the product of the attention and classification layer activations,

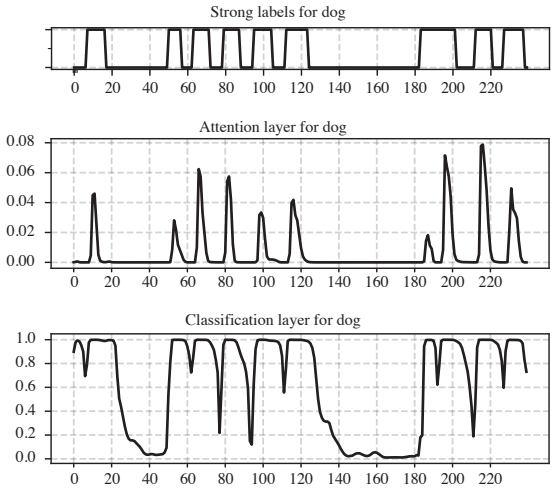


Figure 2: Classification and attention layer activations for file: *Y0a8RB5eOGJ4_30.000_40.000.wav* and class dog.

thresholded with a fixed value for all classes, e.g. 0.5, gives unsatisfactory results. Also it has been shown in similar weakly labelled SED settings that the trained network adapts differently for different classes [10]. Especially there seems to be a difference between classes which tend to have short event durations in contrast to classes which span the majority of timesteps of a clip. Considering this, it might be necessary to use a different post-processing for each class on the networks attention output to account for that. The fact that no strong event annotations are available for training makes this a non-trivial problem, otherwise a simple approach would be to test several post-processing methods and select for each class the one which gives the best performance.

2.4. Self-adaptive label refinement (SALR)

We introduce self-adaptive label refinement, where we check the alignment between strong and weak predictions, and use this as an approximate prediction how well a given post-processing method performs at extracting the right onset and offset of events. Using this approach we can use unlabelled data to estimate how well a given post-processing parameterization performs for each class, and take the best performing parameterization for our final strong prediction.

For post-processing we threshold the output value of the classification layer, followed by a median filter. Therefore the parameters we vary in each iteration are the threshold, and the width of the median filter. However it should be noted that many other methods for post-processing are possible, e.g. a second neural network which maps between the attention layer of the first network and strong predictions might be a potential approach.

In particular, when training has finished, self-adaptive label refinement repeats the following steps on each class with different post-processing parameterizations:

1. A full forward pass is performed to create weak and strong predictions for each clip. Whereby the following steps are only carried out for clips where the weak prediction indicates occurrence of the current class.
2. Using the strong predictions, the spectrogram of each clip is split up into two groups of new samples.

   Each single event occurrence of the examined class is extracted into new samples containing only the temporal frames of the spectrogram which possibly contains the event. Those new samples are labelled with 1.

   Additionally, another sample is created which contains only the temporal frames of the original spectrogram where no occurrence of the given class was predicted. Those are all labelled to 0.
3. The generated new samples are then passed through the network. Using the resulting weak predictions and the labels assigned before, a crossentropy loss for each class is calculated. This loss indicates how good the weak and strong predictions align.

Afterwards for each class, the post-processing with the smallest loss value is selected.

This approach does not need any labels, neither strong nor weak. Therefore our method for post-processing selection is applicable using data of both, the weakly-supervised and the unsupervised dataset. Also the method can be used to adapt the post-processing at inference time to new unseen data.

2.5. Training

The cross entropy loss between the predicted probabilities for each class and the weak ground truth labelling over all labelled clips in a batch is calculated as follows:

$$E = -\frac{1}{N}\sum_{i}^{N}\sum_{c}^{M} l_c^{(i)} log(y_c^{(i)}), \tag{5}$$

where the number of classes is denoted by M, the number of weakly labelled 10 second audio clips by N, $y_c^{(i)}$ denotes the predicted probability for class c of sample i, and $l_c^{(i)}$ is the given binary label in the weakly labelled train set.

In each step a batch containing an equal distribution of samples from the labelled and unlabelled in-domain set is processed. The total loss consists of the cross entropy loss of the labelled samples, regularized with VAT depending on both the labelled and unlabelled samples weighted by a factor λ:

$$\begin{aligned} L = &-\frac{1}{N}\sum_{i}^{N}\sum_{c}^{M} l_c^{(i)} log(y_c^{(i)}) \\ &+ \lambda \sum_{i}^{N'} \mathrm{KL}[p(\mathbf{y}|\mathbf{x}^{(i)};\theta)||p(\mathbf{y}|\mathbf{x}^{(i)}+\mathbf{r};\theta)], \end{aligned} \tag{6}$$

where N' denotes the sum of labelled and unlabelled in-domain clips in a batch, $\mathbf{x}^{(i)}$ is the log-scaled mel-spectrograms of a labelled or unlabelled in-domain clip with index i.

The loss was optimized using Adam [11] with a learning rate of 0.001 and a batch size of 30. The network was implemented using tensorflow [12].

3. EXPERIMENTS AND RESULTS

3.1. Dataset

The method is evaluated using a subset of the Google Audioset [13], which was provided with task 4 of the DCASE 2018 challenge[14].

			no VAT						VAT					
	challenge baseline		no refinement		$SALR_{train}$		$SALR_{dev.}$		no refinement		$SALR_{train}$		$SALR_{dev.}$	
Class	**F1**	**ER**	**F1**	**ER**	**F1**	**ER**	**F1**	**ER**	**F1**	**ER**	**F1**	**ER**	**F1**	**ER**
Alarm bell	3.2%	-	27.0%	1.45	22.4%	1.18	18.8%	1.23	**27.9%**	1.38	21.0%	1.14	18.2%	**1.12**
Blender	15.4%	-	18.5%	2.65	10.7%	1.25	26.9%	1.23	29.9%	1.52	23.2%	1.33	**38.1%**	**0.97**
Cat	0.0%	-	9.5%	3.27	5.0%	1.40	**33.5%**	1.35	4.9%	2.87	19.2%	1.54	25.2%	**1.30**
Dishes	0.0%	-	5.6%	1.65	0.0%	1.16	0.0%	1.16	29.3%	1.93	**32.5%**	**1.16**	**32.5%**	**1.16**
Dog	0.0%	-	**20.5%**	2.16	18.5%	1.40	18.6%	1.39	7.4%	2.00	2.3%	**1.36**	15.8%	**1.36**
Elec. Shaver	32.4%	-	18.4%	2.86	**50.0%**	**0.86**	**50.0%**	**0.86**	14.1%	2.61	40.0%	0.96	40.0%	0.96
Frying	31.0%	-	20.4%	4.54	**43.5%**	1.62	42.9%	1.67	18.0%	3.79	40.0%	1.50	40.7%	**1.46**
Running water	11.4%	-	17.5%	1.86	**37.7%**	1.00	38.0%	**0.99**	22.6%	1.89	31.1%	1.22	32.4%	1.21
Speech	0.0%	-	36.5%	1.38	**44.6%**	**0.95**	36.2%	1.15	37.5%	1.25	41.3%	0.97	40.2%	0.98
Vac. cleaner	46.5%	-	20.0%	3.11	48.8%	1.17	46.5%	1.28	21.8%	2.58	40.5%	1.31	**63.0%**	**0.75**
	14.06%	1.54	19.4%	2.49	28.12%	1.19	31.2%	1.23	21.3%	2.18	29.1%	1.25	**34.6%**	**1.12**

Table 1: **Class-wise results** on the **development set**, total scores are macro averaged.

The majority of the provided audioclips are 10 seconds long, a few audioclips are slightly shorter, for further processing we zero-pad those to a length of 10 seconds. Each audioclip contains one or multiple sound events of 10 different classes, whereby different events may overlap. The dataset consists of a training, testing and evaluation subset.

The training subset consists of 1,578 weakly labelled clips, an unlabelled in-domain set of 14,412 clips and an unlabelled out-of-domain set of 39,999 clips extracted from classes that are not considered in task 4.

The test set contains 288 clips, whereby the distribution in terms of clips per class is similar to the weakly labelled training set. For the test set strong labels from human annotators are given, therefore timestamps for the onset and offset of each event in the clip are included. For training only weak labels are used. The weak labels indicate if a given event occurs somewhere in a 10s clip, however no information about the onset and offset of the events, nor how often the event occurs is given. This setting can also be considered as a multiple instance learning (MIL) problem [10].

Log-scaled mel-spectrograms of each clip are passed as input to the network, for calculation the librosa library [15] is used. Before the spectrograms are calculated, each clip is converted to a mono signal with a sampling rate of 16,000 Hz. For calculation of the log-scaled mel-spectrograms a hamming window of length 1024 with an overlap of 360 is used, this gives 240 frames with 64 mel frequency channels for each clip.

3.2. Baseline system

The organizers of the DCASE challenge provided a baseline system for task 4 [1]. The system consists of two models based on the same structure: three convolution layers with 64 filters of size 3×3, each one followed by a max pooling layer of size 4×1 and a dropout layer with $p = 30\%$. After the convolutional layers, one bidirectional recurrent layer with 64 GRU units and 30% dropout on the input is placed. For output, the first model uses a dense layer with 10 sigmoid units and global average pooling across frames to make clip-level predictions, and the second model uses a time distributed dense layer with 10 sigmoid units to predict events at frame-level. Training of the system is performed in two steps:

1. The first model is trained with the weakly labelled training set, then the trained model is used to generate weak labels for the unlabelled in-domain set.
2. The second model is trained on the unlabelled in-domain set, using the weak labels generated beforehand. In this second training pass the weakly labelled set is used for validation.

As input, each 10 second audio file is divided into 500 frames of 64 log mel-band magnitudes.

3.3. Evaluation

For evaluation the macro averaged event-based F-score [16] is used. The event-based metrics are calculated using the open source toolbox sed_eval [17]. As given by the challenge, for calculation of event-based metrics a 200ms collar on onsets and a 200ms / 20% of the events length collar on offsets was set. For calculation of the total performance over all individual classes, macro averaging is used. This has the effect that each class has equal influence on the final metrics, even if the distribution of classes in the tested set is unbalanced.

3.4. Results

Table 1 shows the event based F1 scores and error rates of our system on the development set. We compare the resulting scores of our system without post-processing refinement, and when we performed self-adaptive label refinement using data either of the training or the development set. Additionally, we also show the influence of VAT. When no post-processing refinement was done, we calculated the strong labels with a fixed threshold of 0.5 for all classes and apply no median filter. It can be seen that both SALR and VAT increase the performance of the system. Whereby when SALR is used, the best performance is achieved when the adaption was done on the development set.

3.5. Submitted systems

Three systems have been submitted to the DCASE 2018 challenge, whereby self-adaptive label refinement was used to adapt the post-processing as follows: System one has been adapted to the evaluation set. System two did not use any adaption, but used the same post-processing with a fixed threshold of 0.5 and a median filter width of 1. System three has been adapted to the training set.

4. CONCLUSION

In this paper, we proposed a method for sound event detection using only weakly labelled and unsupervised data. Our approach is based on GCRNNs, whereby we introduce self-adaptive label refinement. This method adapts the postprocessing using unlabelled data, and increases SED performance. The final F-score of our system is 34.6%, which is significantly higher than the score of the baseline system which is 14.06%.

5. REFERENCES

[1] R. Serizel, N. Turpault, H. Eghbal-Zadeh, and A. Parag Shah, "Large-Scale Weakly Labeled Semi-Supervised Sound Event Detection in Domestic Environments," in *Workshop on Detection and Classification of Acoustic Scenes and Events*, Woking, United Kingdom, Nov. 2018, submitted to DCASE2018 Workshop. [Online]. Available: https://hal.inria.fr/hal-01850270

[2] Y. Xu, Q. Kong, W. Wang, and M. D. Plumbley, "Large-scale weakly supervised audio classification using gated convolutional neural network," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2018, Calgary, AB, Canada, April 15-20, 2018*, 2018, pp. 121–125. [Online]. Available: https://doi.org/10.1109/ICASSP.2018.8461975

[3] T. Miyato, S. Maeda, S. Ishii, and M. Koyama, "Virtual adversarial training: A regularization method for supervised and semi-supervised learning," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–1, 2018.

[4] T. Miayto, A. M. Dai, and I. Goodfellow, "Virtual adversarial training for semi-supervised text classification," 2016.

[5] M. Zöhrer and F. Pernkopf, "Virtual adversarial training and data augmentation for acoustic event detection with gated recurrent neural networks," pp. 493–497, 08 2017.

[6] M. Ratajczak, S. Tschiatschek, and F. Pernkopf, "Virtual adversarial training applied to neural higher-order factors for phone classification," in *Interspeech*, 2016.

[7] A. Oliver, A. Odena, C. Raffel, E. D. Cubuk, and I. J. Goodfellow, "Realistic evaluation of semi-supervised learning algorithms," 2018. [Online]. Available: https://openreview.net/forum?id=ByCZsFyPf

[8] Y. N. Dauphin, A. Fan, M. Auli, and D. Grangier, "Language modeling with gated convolutional networks," *CoRR*, vol. abs/1612.08083, 2016. [Online]. Available: http://arxiv.org/abs/1612.08083

[9] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997. [Online]. Available: https://doi.org/10.1162/neco.1997.9.8.1735

[10] B. McFee, J. Salamon, and J. P. Bello, "Adaptive pooling operators for weakly labeled sound event detection," *IEEE/ACM Trans. Audio, Speech and Lang. Proc.*, vol. 26, no. 11, pp. 2180–2193, Nov. 2018. [Online]. Available: https://doi.org/10.1109/TASLP.2018.2858559

[11] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," 12 2014.

[12] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, *et al.*, "Tensorflow: a system for large-scale machine learning."

[13] J. F. Gemmeke, D. P. Ellis, D. Freedman, A. Jansen, W. Lawrence, R. C. Moore, M. Plakal, and M. Ritter, "Audio set: An ontology and human-labeled dataset for audio events," in *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*. IEEE, 2017, pp. 776–780.

[14] http://dcase.community/workshop2018/.

[15] B. McFee, C. Raffel, D. Liang, D. P. Ellis, M. McVicar, E. Battenberg, and O. Nieto, "librosa: Audio and music signal analysis in python," in *Proceedings of the 14th python in science conference*, 2015, pp. 18–25.

[16] A. Mesaros, T. Heittola, and T. Virtanen, "Metrics for polyphonic sound event detection," *Applied Sciences*, vol. 6, no. 6, p. 162, 2016. [Online]. Available: http://www.mdpi.com/2076-3417/6/6/162

[17] A. Heittola, T.; Mesaros, "sed_eval - evaluation toolbox for sound event detection." https://github.com/TUT-ARG/sed_eval, accessed: 2018-07-20.

ENSEMBLE OF CONVOLUTIONAL NEURAL NETWORKS FOR GENERAL-PURPOSE AUDIO TAGGING

Bogdan Pantic

School of Electrical Engineering
Signals and Systems Department
Belgrade, Serbia
bogdan.pantic@yahoo.com

ABSTRACT

This work describes our solution for the general-purpose audio tagging task of DCASE 2018 challenge. We propose the ensemble of several Convolutional Neural Networks (CNNs) with different properties. Logistic regression is used as a meta-classifier to produce final predictions. Experiments demonstrate that the ensemble outperforms each CNN individually. Finally, the proposed system achieves Mean Average Precision (MAP) score of 0.945 on test set, which is a significant improvement compared to the baseline.

***Index Terms*—** audio tagging, DCASE 2018, convolutional neural networks, ensembling

1. INTRODUCTION

The goal of audio tagging is to create models capable of recognizing a variety of sounds. Those include musical instruments, vehicles, animals, sounds generated by some sort of human activity etc. The motivation for a research in the field of an artificial sound understanding can be found in potential applications such as security, healthcare (hearing impairment), improvements in smart devices, various music related tasks etc. Detection and Classification of Acoustic Scenes and Events (DCASE) challenge 2018 consists of several tasks which provide the way to evaluate different methods for solving problems related to non-speech audio signals. The focus of this paper will be on the task 2: "General-purpose audio tagging of FreeSound content with AudioSet labels" which is hosted on Kaggle platform [1]. The dataset contains around 9500 training and 1600 testing examples which belong to one of 41 unequally distributed classes (bus, gunshot, knock, flute, etc.). Audio files differ in length with the duration ranging from 300ms to 30s. All samples were automatically annotated, but only a portion of training set labels were manually verified. Therefore, there is a large variation in label quality which poses yet another problem to participants – to extract as much information as possible from the weakly labeled data. Another major issue is a label density. It represents a portion of audio in which the tagged event is actually present. As one can imagine, the label density can vary significantly, so creating models which can successfully tackle that is of high importance.

Though the research in this area has recently expanded, a related work can be found at the previous editions of DCASE challenge. Alternatively, a related research can also be found in the area of Music Information Retrieval (MIR). The earlier research mostly relied on hand crafted features and shallow models. For example, in the first edition of DCASE in 2013 models like SVM [2] and bagging of decision trees [3] were used with the variety of features. Similar tendency can be found in the MIR research where features were particularly designed to capture timbral and rhythmic characteristics [4]. The later research shows an obvious shift towards feature learning, more precisely, deep learning. Following their success in computer vision, convolutional neural networks (CNN) are extensively used for the audio scene classification [5, 6], event detection [7, 8], music tagging [9] etc. One can use CNNs in different settings and on different input representations. Using raw audio with one-dimensional convolutions is a viable option, but most research relies on some sort of time-frequency representation and two-dimensional CNNs as it is typically expressive enough and less computationally expensive. Mel-spectrograms are widely used, but Constant Q transform (CQT) also shows promising results [10]. Although a computer vision inspired the rapid growth of CNN usage, the interpretation in audio domain fundamentally differs. While vertical and horizontal axes in images generally satisfy the same properties and should be treated equivalently, time and frequency axes of an audio signal represent different modalities. Therefore, there is a room for the domain specific filter design, which should capture interesting patterns, improve CNN architecture efficiency and hopefully increase model performance. Several researchers have already tried to exploit these facts, yielding competitive results in related areas. Depending on a problem in question, approaches focus on modelling temporal [11] and frequency [12] related features with horizontal and vertical filters respectively. Especially interesting for our dataset are wide architectures that incorporate parallel feature learning [13, 14]. In that setting, one should be able to use many different filter shapes and fuse extracted features in later layers, which would enable the model to learn much richer set of descriptors. Due to the nature of our problem, mainly, large differences in acoustic properties of provided classes, parallel architectures could prove to be beneficial.

This paper describes our solution for DCASE challenge. We will evaluate several architectures and preprocessing techniques. The main goal is to design diverse set of classifiers and leverage these differences by stacking predictions of individual models. The paper is organized as follows. Validation, preprocessing, proposed models and ensembling are described in the section 2. Section 3 deals with the evaluation and details of experimental setup. Finally, the obtained results are presented in the section 4.

2. SYSTEM ARCHITECTURE

This section provides an overview of the most crucial aspects of the solution, including validation setup, data preprocessing, CNN

architectures and ensembling technique.

2.1. Validation Setup

One of the major decisions during the development of the machine learning system is a configuration of the train-validation split. It is common to use K-fold cross-validation, where a model is trained on K-1 folds and validated on the remaining one. At the end, the average score is used as a performance estimate. However, in the case of the provided dataset, there is a large percent of samples which are not manually verified in the training set and none of them in the test set. Since validation should represent unseen data as close as possible, we choose to use only manually verified examples. The same split is used for all models, where 10% of the data is used for validation. It is also worth mentioning that train and validation data have the same distribution of labels, but the distribution of manually verified samples of different classes in the training set is not uniform.

2.2. Preprocessing

In the introduction, it is pointed out that audio files have different length and consequentially don't contain the same amount of information. Therefore, the preprocessing should account for both fixed size input requirement of our models and the information perseverance. The input audio length is predefined, and it varies between 4 and 8 seconds for different models while shorter files are zero padded. The sampling rate is 44.1 kHz. Two representations are used: mel-spectrogram with 96 mel bands and constant Q transform with 96 or 110 bands. Longer files are split into chunks of predefined length with several overlap values and the resulting spectrograms are converted to dB-scale (the amplitude is scaled relative to maximum value). Obtained inputs are of shape (n_{bands}, n_{frames}) and they are all used as new training examples (which means that the resulting dataset is larger than the initial one). Finally, the data is standardized (by subtracting the mean and dividing by the standard deviation of the entire training set). At the test time, identical transformations are applied and the results are generated as an average of the predictions of each chunk corresponding to the same file. The entire preprocessing is done using Librosa library [15], and the remaining hyper-parameters are left to the default values.

2.3. Network Architectures

The ensembling is known to yield the highest benefits when predictions of the base models are less correlated. To fully exploit that fact, we propose a couple of architectures with slightly different properties.

The first network is inspired by one of the top solutions of "Tensorflow Speech Recognition challenge" [16] and suggested by other participant of DCASE challenge [17]. The initial convolutional layer has 64 filters of shape 7x3, followed by 4x1 max-pooling layer. The next layer contains 128 filters of shape 7x1 and 4x2 max-pooling with 2x2 strides. Finally, two convolutional layers with 128 filters and 1x5 and 5x1 shapes respectively are stacked before the global-max-pooling layer. Two densely-connected layers with 64 neurons are used for an additional feature extraction before a softmax classifier. The activation function of each layer is rectified linear unit and each convolutional layer is followed by batch-normalization. Dropout of 0.25 is used before each dense layer for additional regularization.

The second architecture is proposed in [13]. It relies heavily on the domain knowledge, by introducing sets of rectangular filters applied in parallel on input. On the one hand, for frequency related features, vertical filters which cover 90% and 40% of domain are used with small temporal dimension. On the other hand, to capture temporal features efficiently, an average pooling is applied over frequency axis of spectrogram and several 1D convolutional kernels are employed. The filter lengths are 165, 128, 64 and 32. Outputs of both frequency and time related feature extractors are concatenated. Three 2D convolutional layers with 512 filters are then applied, the result is flattened and the dense layer with 300 neurons followed by 0.4 dropout is added before the output layer.

Additionally, to explore other aspects of rectangular filter design, a new architecture is proposed. It is inspired by [12] and the details are given in the table 1:

Table 1: Description of model 3: Set of rectangular filters, concatenation of feature maps and additional layers

Conv1: 48x (8x7)| 32x (32x7)| 16x (64x7)| 16x (90x7) + BN
Concatenate
Max-Pooling (5x5)
Conv2: 120x (2x2)
Global-Max-Pooling 2D
Dense1: 64 units + Dropout 0.2
Dense2: 64 units + Dropout 0.2
Dense3: 41 units + softmax

The output of each branch in *Conv1* layer has to be the same, so that feature maps can be stacked. This is achieved by zero-padding input accordingly. All hidden layers use rectified linear unit as activation.

The combination of previously discussed ideas has led to yet another model:

Table 2: Description of model 4: Set of rectangular filters with more depth, concatenation of feature maps, additional convolutional and dense layers

Conv1: 64 x (8x3)| 64 x (16x3)| 64 x (32x3)
Max1:Max-Pooling (4x1) + BN
Conv2: 128 x (8x1)| 128 x (16x1)| 128 x (32x1)
Max2: Max-Pooling (4x2) + BN
Concatenate
Conv3: 128 x (5x1) + BN
Conv4: 128 x (1x5) + BN
Global-Max-Pooling 2D
Dense1: 64 units + Dropout 0.2
Dense2: 64 units + Dropout 0.2
Dense3: 41 units + softmax

The fourth model is using architectural designs of the first network, but with the parallelism introduced in the models two and three. Instead of the single convolutional layer before the concatenation, it is using two layers per branch. Same padding is used in these two layers to avoid a dimensionality mismatch. The strides of max-pooling layers are 2x1 and 2x2 respectively. Similarly, hidden layers use ReLU activation.

Models which are typically used in a computer vision community can be added to maximize diversity. Concretely, we use

Inception V3 [18] and MobileNet [19] with weights pretrained on ImageNet. They are implemented using Keras [20] library. The classification layer is removed from both architectures and two layers with 64 units and 0.2 dropout are added before the softmax layer with 41 units. The additional preprocessing steps are required for these setups. We had to resize the inputs to 150x150 for Inception and 160x160 for MobileNet to match implementation requirements. Also, a number of channels had to be matched, so mean is calculated across the entire training data and added as the second and the third channel to each sample. These models require more computing time, but add a significant value to ensembles. We will refer to Inception as model 5 and MobileNet as model 6 in the remainder of the paper.

2.4. Ensembling

Once models are configured and trained there are many ways to leverage generated predictions. Calculating arithmetic or geometric mean are two obvious ways, since they don't add additional complexity and almost always improve performance. However, once we have a sufficiently diverse set of base predictions, real gains come with stacking. Stacking is performed by using predictions as features for a meta-model. It is often done in a cross-validation setting, but because of the train-validation split used by level-1 models, we are constrained to use only 10% of data for meta-model training. The predictions of the individual classifiers are stacked in the columns for both the validation and the test set. For example, each of ten models would have 41 (number of classes) predictions per example and the resulting feature matrix would have 410 columns. The validation data then becomes a new training set and stratified 5-fold cross-validation is used for training. The experiments have shown that logistic regression is suitable candidate for the meta-classifier. Principal component analysis (PCA) is used for a dimensionality reduction in each of K iterations. The predictions of the meta-model on the validation data are combined and MAP score is computed to produce a new performance estimate. Finally, trained meta-model is used to generate test set predictions.

3. EXPERIMENTAL DESIGN

3.1. Evaluation

Organizers split test data in a public and a private part. Participants submit their predictions for the entire test set, but they can only see a public score (contains around 19% of test data). Submissions are evaluated using mean average precision:

$$MAP = \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{min(n,3)} P(j) \quad (1)$$

where N is a number of audio files used for scoring, n is a number of predictions per file and $P(j)$ is the precision at cutoff j. The private score is released after the competition ends.

3.2. Hyper-parameters and data augmentation

The input is split into patches of length 4s, 5s or 6s with 1s overlap, or 8s with 2s overlap. The experiments on shorter inputs gave worse scores and didn't add any value to the ensembles, so they were discarded. The networks were trained using categorical cross-entropy as a loss function. Adam is used as an optimizer, with the initial learning rate of 0.001. The mini-batch size was 32 or 64, depending on a model and the input size. During training, we monitor a validation loss and save currently the best performing model. If the validation loss doesn't decrease for seven epochs, the learning rate is multiplied by 0.5. Early-stopping is used to avoid overfitting. The training stops after 20 epochs passed from the last improvement. A maximum number of epochs for all models is 250.

Data augmentation is another way to reduce overfitting. Transformations are applied to the original data points, artificially enlarging dataset. It's crucial that augmentation techniques do not change a true label of the particular sample, otherwise performance may decrease. Concretely, we used random width shift and zoom with maximum range of 0.1. Another interesting augmentation technique which significantly reduced overfitting is random erasing [21]. It works by randomly selecting rectangular area on the input image and changing its values with random numbers. Finally, the most important augmentation technique used is mixup [22]. It is implemented by creating virtual feature-target pairs $(\tilde{x}, \tilde{y})$:

$$\tilde{x} = \lambda x_i + (1-\lambda)x_j \quad (2)$$
$$\tilde{y} = \lambda y_i + (1-\lambda)y_j \quad (3)$$

where (x_i, y_i) and (x_j, y_j) are the pairs drawn randomly (regardless of the provided label of the sample) from the training data, while $\lambda \sim Beta(\alpha, \alpha)$. Therefore, the parameter α affects a regularization strength (larger α implies stronger regularization). Mixup is encouraging linear behavior between training examples which has other positive side effects. The original paper shows that it also improves robustness to corrupted labels. They argue that by increasing α it should be possible to create virtual examples further from the original, wrongly labeled samples and therefore reduce effect of the memorization of the corrupted classes. As discussed previously, the majority of the training examples were not manually verified (accuracy of those labels is estimated to be at least 65-70% per class), so mixup allows us to decrease negative impact of the label noise with minimal additional computational requirements, since each "new" training example is just a linear combination of two original samples. These desirable properties have made mixup a crucial part of every pipeline. Regarding the parameter value, α between 0.2 and 0.3 was found to be optimal across different architectures. Every augmentation technique is performed during the training phase.

4. RESULTS

In this section, the results of the proposed architectures are presented and discussed. Table 3 summarizes important information regarding the models and their scores on a test set (public and private part combined). For clarity, models are specified only through ids, with respect to the order of presentation. These particular combinations of the models and the preprocessing techniques have been chosen based on the estimated performance of the ensemble on the validation data. Additionally, to avoid the usage of too many configurations in the final ensemble, the models

which have brought only small improvements haven't been included. We can come up with several interesting conclusions by inspecting these values. The experiments have shown that CQT outperforms mel-spectrogram for most models. Nevertheless, mel-spectrograms have added significant diversity, so they have been kept for the stacked ensemble. Also, the inputs of length 4s and 5s seem to be optimal, since they provide nice trade-off between performance, required computation and the memory usage. The larger number of bands certainly helps, but it also increases computation time.

Organizers provided a baseline model for comparison with the proposed solutions. Its inputs are log scale mel-spectrograms with windows of length 0.25s and hop size of 0.125s. The model contains 3 convolutional layers with filter shapes 7x7, 5x5 and 3x3 respectively before the output softmax layer. Total number of parameters is around 658.1K. It achieves MAP score of approximately 0.70 on test set (0.7 on public and 0.69 on private leaderboard).

Table 3: Summary of architectures: Models, length of audio chunks, overlap used for longer files, input representations, MAP scores

Id	Length	Overlap	Transform	Bands	MAP
1	4s	1s	Mel-spec	96	0.87
1	5s	1s	CQT	110	0.913
1	6s	1s	Mel-spec	96	0.878
1	8s	2s	CQT	110	0.909
2	4s	1s	CQT	96	0.915
2	5s	1s	Mel-spec	96	0.889
3	4s	1s	CQT	96	0.872
4	4s	1s	CQT	96	0.908
5	4s	1s	CQT	96	0.895
5	5s	1s	CQT	96	0.894
6	4s	1s	CQT	110	0.909

Final solution is an ensemble of 11 proposed configurations. The meta-model is logistic regression with the regularization parameter $C = 4$. It is trained on 120 features, after PCA dimensionality reduction. The average precision scores per class (taking into account only top three predictions per example, referred to as AP@3) are shown in Table 4. Considering high overall performance of both low level classifiers and the ensemble, the results are expected, since most of the classes obtained nearly perfect scores. However, there are a few exceptions, most notably: "Squeak", "Fireworks" and "Scissors". "Squeak" is a class which has one of the lowest percentages of the manually verified examples, which could be a reason for the bad score. "Fireworks" are, intuitively, often confused with gunshots, which seems natural and it is something that would be problematic even for a human listener. Finally, the class "Scissors" has the 2nd smallest number of samples in the training set and the smallest in the test set. There are a couple of less problematic classes like "Chime" and "Glockenspiel" (often confused), "Gunshot, gunfire" (same as fireworks), "Bus" and "Telephone".

The proposed ensemble achieves MAP of 0.956 on the public leaderboard, 0.942 on the private leaderboard and 0.945 on the entire test set, which is an improvement over level-1 models and the baseline. The final submission ranked 12th among 558 competing teams on the private leaderboard.

5. CONCLUSION

This paper proposes an ensemble of convolutional neural networks for the classification of general audio signals. We have introduced several architectures, preprocessing techniques and validation setup in order to get a diverse set of base predictions. Logistic regression is then used as a meta-model to obtain the final output. It has been shown that it outperforms individual models substantially, which demonstrates that original architectures really provide sufficiently diverse information. Further improvements might be possible by including different non-deep learning models with hand crafted features, additional data augmentation or by adding more pre-trained models to the ensemble.

Table 4: Per-category results: Class, number of samples and AP@3

Class	Samples	AP@3
Acoustic guitar	45	0.93
Applause	32	1.0
Bark	28	0.964
Bass drum	28	1.0
Burping, eructation	32	1.0
Bus	25	0.873
Cello	54	0.981
Chime	29	0.896
Clarinet	56	0.988
Computer keyboard	26	0.904
Cough	30	1.0
Cowbell	42	1.0
Double bass	40	0.987
Drawer open, close	29	0.931
Electric piano	32	0.969
Fart	30	0.983
Finger snapping	33	1.0
Fireworks	32	0.76
Flute	55	0.972
Glockenspiel	29	0.868
Gong	37	1.0
Gunshot, gunfire	63	0.889
Harmonica	33	0.955
Hi-hat	39	0.949
Keys jangling	28	0.946
Knock	39	0.957
Laughter	38	0.947
Meow	29	1.0
Microwave oven	29	0.983
Oboe	42	0.96
Saxophone	110	0.958
Scissors	25	0.78
Shatter	29	0.983
Snare drum	34	0.917
Squeak	29	0.672
Tambourine	40	0.95
Tearing	27	0.962
Telephone	48	0.809
Trumpet	37	0.946
Violin, fiddle	108	0.995
Writing	29	0.943

6. REFERENCES

[1] E. Fonseca, M. Plakal, F. Font, D. P. W. Ellis, X. Favory, J. Pons, X. Serra, "General-purpose Tagging of Freesound Audio with AudioSet Labels: Task Description, Dataset, and Baseline," Submitted to *DCASE 2018 workshop*, 2018.

[2] J.T. Geiger, B. Schuller, G. Rigoll, "Recognizing acoustic features with large-scale audio feature extraction and SVM," *IEEE AASP Challenge on Detection and Classification of Acoustic Scenes and Events*, 2013.

[3] D. Li, J. Tam, D. Toub, "Auditory scene classification using machine learning techniques," *IEEE AASP Challenge on Detection and Classification of Acoustic Scenes and Events*, 2013.

[4] G. Tzanetakis, P. Cook, "Musical genre classification of audio signals*," IEEE Transactions on Speech and Audio Processing*, vol. 10, no. 5, July 2002.

[5] M. Valenti, A. Diment, G. Parascandolo, S. Squartini, T. Virtanen, "DCASE 2016 acoustic scene classification using convolutional neural networks," *Detection and Classification of Acoustic Scenes and Events*, 2016.

[6] S. Park, S. Mun, Y. Lee, H. Ko, "Acoustic scene classification based on convolutional neural network using double image features," *Detection and Classification of Acoustic Scenes and Events*, 2017.

[7] D. Lee, S. Lee, Y. Han, K. Lee, "Ensemble of convolutional neural networks for weakly supervised sound event detection using multiple scale input," *Detection and Classification of Acoustic Scenes and Events*, 2017.

[8] I. Jeong, S. Lee, Y. Han, K.Lee, "Audio event detection using multiple-input convolutional neural network," *Detection and Classification of Acoustic Scenes and Events*, 2017.

[9] K. Choi, G. Fazekas, M. Sandler, "Automatic tagging using deep convolutional neural networks," *International Society of Music Information Retrieval Conference*, 2016.

[10] T. Lidy, A. Schindler, "CQT-based convolutional neural networks for audio scene classification and domestic audio tagging," *Detection and Classification of Acoustic Scenes and Events*, 2016.

[11] J. Schluter, S. Bock, "Improved musical onset detection with convolutional neural networks," *IEEE International Conference on Acoustic, Speech and Signal Processing*, 2014.

[12] E. Fonseca, R. Gong, D. Bogdanov, O. Slizovskaia, E. Gomez, X. Serra, "Acoustic scene classification by ensembling gradient boosting machine and convolutional neural networks," *Detection and Classification of Acoustic Scenes and Events*, 2017.

[13] J. Pons, O. Nieto, M. Prockup, E. Schmidt, A. Ehmann, X. Serra, "End-to-end learning for music audio tagging at scale," in *Proceedings of International Society for Music Information Retrieval Conference*, 2018.

[14] A. Schindler, T. Lidy, A. Rauber, "Multi-temporal resolution convolutional neural networks for the DCASE acoustic scene classification task," *Detection and Classification of Acoustic Scenes and Events*, 2017.

[15] https://librosa.github.io/librosa/

[16] https://www.kaggle.com/c/tensorflow-speech-recognition-challenge/discussion/47715

[17] https://www.kaggle.com/c/freesound-audio-tagging/discussion/57051

[18] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, Z. Wojna, "Rethinking the inception architecture for computer vision," *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[19] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, "MobileNets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv: 1704.04861*, 2017.

[20] https://keras.io/

[21] Z. Zhong, L. Zheng, G. Kang, S. Li, Y. Yang, "Random erasing data augmentation," *arXiv preprint arXiv: 1708.04896*, 2017.

[22] H. Zhang, M. Cisse, Y.N. Dauphin, D. Lopez-Paz, "Mixup: Beyond empirical risk minimization," *International Conference on Learning Representations*, 2018.

SAMPLE MIXED-BASED DATA AUGMENTATION FOR DOMESTIC AUDIO TAGGING

Shengyun Wei[1], *Kele Xu*[1,2], *Dezhi Wang*[3], *Feifan Liao*[1], *Huaimin Wang*[2], *Qiuqiang Kong*[4],

[1] National University of Defense Technology, Information and Communication Dept., Wuhan, China, yumocloud@protonmail.ch, liaofeifan@126.com
[2] National University of Defense Technology, Computer Dept., Changsha, China, kelele.xu@gmail.com, whm_w@163.com
[3] National University of Defense Technology, College of Meteorology and Oceanography, Changsha, China, wang_dezhi@hotmail.com
[4] University of Surrey, Center for Vision, Speech and Signal Processing, Guildford, UK, q.kong@surrey.ac.uk

ABSTRACT

Audio tagging has attracted increasing attention since last decade and has various potential applications in many fields. The objective of audio tagging is to predict the labels of an audio clip. Recently deep learning methods have been applied to audio tagging and have achieved state-of-the-art performance, which provides a poor generalization ability on new data. However due to the limited size of audio tagging data such as DCASE data, the trained models tend to result in overfitting of the network. Previous data augmentation methods such as pitch shifting, time stretching and adding background noise do not show much improvement in audio tagging. In this paper, we explore the sample mixed data augmentation for the domestic audio tagging task, including mixup, SamplePairing and extrapolation. We apply a convolutional recurrent neural network (CRNN) with attention module with log-scaled mel spectrum as a baseline system. In our experiments, we achieve an state-of-the-art of equal error rate (EER) of 0.10 on DCASE 2016 task4 dataset with mixup approach, outperforming the baseline system without data augmentation.

Index Terms— Audio tagging, data augmentation, sample mixed, convolutional recurrent neural network

1. INTRODUCTION

Audio tagging is to label each audio recording with one or more of a multi-label set of labels. This task has many applications such as audio surveillance [1], recommendation system [2] and animal populations monitoring [3], where determining the presence of events in the acoustic scene is the top priority. Manually tagging audio clips is time-consuming and tedious process with growing amounts of data. Consequently, several audio tagging challenges such as DCASE 2016-2017 [4], [5] have been held in recent years. Deep convolutional recurrent neural network with attention module have achieved the state-of-the-art performance on DCASE 2016 dataset. Due to the size of many audio tagging datasets are limited to hours [6], it remains as a challenge to improve the generalization ability of the network, especially when the training data size is limited. Selection of the model complexity is important for a deep neural network to obtain better generalization performance. For example, by increasing the complexity of a model, the representational ability can be increased, however, it also might increase the possibility of overfitting [7].

To increase the generalization ability of the deep neural networks, sustainable efforts have been proposed. For example, dropout [8] and batch normalization [9] are widely-used regularization techniques for the hidden states of the network. To regularize the intermediate layers in a neural network, several variants have been proposed, such as max-drop and stochastic dropout [10]. Moreover, shake-shake regularization [11] and shake drop regularization decrease error rates by disturbing learning [12].

On the other hand, data augmentation is a crucial component of the state-of-the-art methods[13] for different tasks. For example, Random flipping, random cropping and horizontal flipping are widely-used augmentation approaches for the images. For acoustic modeling, pitch shifting, time stretching, and dynamic range compression extend an audio training set with perturbed samples [14]. However, these simple data augmentation have a negligible impact on the acoustic modeling performance. In this paper, we propose to use sample mixed data augmentation for audio tagging.

An alternative method for data augmentation is to combine the training samples together, which is called sample mixed-based data augmentation in this paper. For image, SamplePairing synthesizes a new sample from one image by overlaying another image randomly chosen from the training data [15]. The label of the mixed sample is the same as the label of the first image, which is considered as label-preserving. Compared with label-preserving methods, mixup provides a better generalization ability for image by mix multiple examples, and mixup has demonstrated surprisingly effectiveness [16]. In this paper, we target to explore the mixup, SamplePairing and extrapolation data augmentation for audio tagging task in DCASE 2016 (Task 4).

This paper is organized as follows. In Section 2, the preprocessing method is given, while Section 3 describes the data augmentation methods, the employed data augmentation methods include mixup and its variants, SamplePairing and sample extrapolation. Section 4 provides a detailed description of our network architecture, and Section 5 gives the experimental results evaluated on the development dataset. Finally, Section 6 summarizes the paper and provides conclusions.

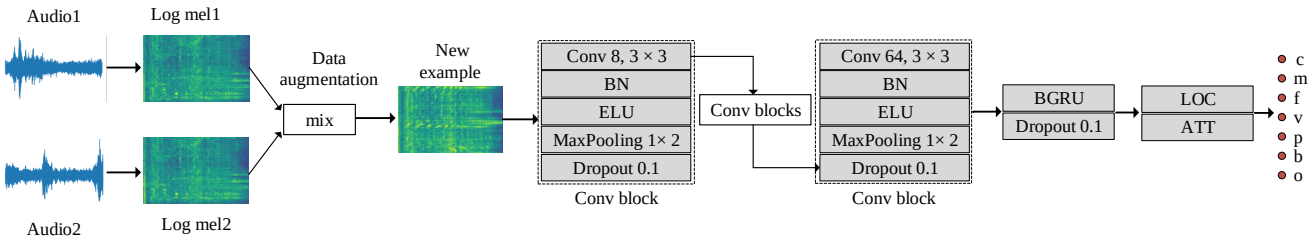


Figure 1: System architecture for audio tagging. Input is log-scaled mel spectrogram with 124 frames and 128 mel frequency bins. Data augmentation is operated on the input samples. There are 7 conv blocks and the number of filters in the convolution layers is 8, 16, 32, 64, 64, 64, 64 respectively.

2. CONVOLUTIONAL RECURRENT NEURAL NETWORK WITH ATTENTION AND LOCALIZATION

Deep convolutional neural networks(CNNs) can provide superior performance for audio tagging [17], [18]. For acoustic models, CNNs have been used extensively compared to fully-connected DNNs. Due to modeling local correlations with CNNs can capture spatial local correlation information effectively. Moreover, CNNs reduce spectral variations within audio signals by replicating weights across time and frequency.

Fig.1 shows the architectures of employed CRNN for the audio tagging task. For our neural network architecture, three main parts are employed: time-data augmentation for the frequency representation in the sample space, feature extraction by CRNN and classification to generate output. Log mel spectrograms with 128 frequency bins are used as the input representation. In the Fourier transform, a Hamming window with size of 1024 is used. A stack of convolutional layers are used to extract robust features. All convolutional layers with small kernel size of 3×3 are followed by a batch normalization layer, a max-pooling layer with size 1×2. A dropout layer with ratio 0.1 is used for prevent overfitting. Each block is called Conv block for short. Exponential linear units (ELUs) is used, due to ELUs lead to faster learning and to better generalization performance than ReLUs [19]. In addition, Binary cross-entropy is used as the loss function, which gives better results than the quadratic cost [20].

Attention and localization based deep convolutional recurrent model (ATT-LOC) achieved the state-of-the-art performance on the evaluation set [21]. Our structure is similar to ATT-LOC, as both including a deep convolutional recurrent model with an attention module and a localization module. The attention mechanism can effectively reduce the impact of background noise on the output and make the classifier to pay more attention to the frame in which the acoustic events occurr. The localization module detects the onset and offset time of acoustic events in the audio chunk. More details of the attention and localization mechanism could be found in [21].

However, some modifications have been made based on the ATT-LOC and the differences between our model and ATT-LOC model include: 1) Different input. The basic features along with the spatial features are concatenated to be fed into the model in ATT-LOC, whereas we just use the basic features of mono audio. This is due to using spatial information of stereo audio to improve results in audio tagging might not always work. The ATT-LOC receives no benefit at all from interaural phase differences and interaural magnitude differences, while only IMD has benefits [21]. It all depends on the context in which sound events occur and their relative location. Furthermore, mono audio is more general and accessible. 2) Different convolutional layers. ATT-LOC has only one convolutional layer with big kernel size of 30×1, On the contrary, inspired by VGG [22], our network have 7 convolutional layers which are equipped with small kernel size of 3×3) In addition, the number of convolutional layers can be changed automatically according to the input. Due to the CNN can extract the features of different levels, the more layers the network has, the richer features of different levels can be extracted. Moreover, the more abstract the features of network extraction are, the more semantic information there is. However, simply increasing the depth can result in gradient dispersion or gradient explosion. In fact, we have tried to use deeper networks, such as VGG, but the results have not been satisfactory. We conjecture that the size of the dataset does not match the model capability.

3. SAMPLE MIXED-BASED DATA AUGMENTATION

3.1. Mixup

Data augmentation aim to expand the training data size by creating new samples, with the goal to reduce the generalization gap between the training and test data. Recently, mixup method has provided better performance for many tasks [23]. In more detail, mixup generates synthetic samples using interpolation in a manner, which is not label-preserving. Unlike the previous attempts to encoder the label using the one-hot encoder approach, the new labels for mixed samples do not belong to two classes, but using the weighted probability of the label. From another perspective, mixup calculates the cross-entropy loss on the two labels with the weighted input, and the two final losses are weighted. The training set can be seen as a bunch of scatters distributed in high-dimensional space. Many new data points between the training set scatter are created by mixup. With the expanded dataset, the relative distance between the scatter points is reduced. To be simpler and more efficient, mixup is applied to a single minibatch and its shuffled version. And the new minibatch can be represented by:

$$\begin{aligned} x_n &= \lambda * x_i + (1-\lambda) * x_j \\ y_n &= \lambda * y_i + (1-\lambda) * y_j \end{aligned} \tag{1}$$

where mixing proportion $\lambda \sim Beta(\alpha, \alpha)$, for $\alpha \in (0, \infty)$, and $\lambda \in [0, 1]$, (x_i, y_i) and (x_j, y_j) are pairs of samples selected from a minibatch.

3.2. SamplePairing

Analogously, SamplePairing [15] synthesize a new sample by taking an average of two inputs vectors. The label of the mixed sample is the same as the first sample. Hence, Sample pairing constructs new training examples as:

$$\begin{aligned} x_n &= 0.5 * x_i + 0.5 * x_j \\ y_n &= y_i \end{aligned} \tag{2}$$

Once new samples have been created by interpolating and pairing between two log mel features, they can be used directly as the input for a deep neural network model.

3.3. Mixup with label preserving(mixup_lp)

Mixup_lp is a combination of mixup and SamplePairing. New samples are generated by the fashion of linear interpolation, while in this way that does not use convex combinations of labels.

3.4. Extrapolation

Extrapolation operator is used to generated useful synthetic examples in feature space [24]. It is worthwhile to note that, sample mixed-based data augmentation can be either interpolation or extrapolation between a pair of samples in input space. Unlike mixup, which only explore the interpolation between two samples, we explore both interpolation and extrapolation exploration for the domestic audio tagging in this paper. In more detail, extrapolation between two samples is a different alternative of linear combination. However, binary cross-entropy error has negative values when generate the labels by extrapolation, due to the categories are labeled as values larger than 1 instead of 0 and 1, which confuses the classifier. So we have:

$$\begin{aligned} x_n &= (1 + \lambda) * x_i - \lambda * x_j \\ y_n &= y_i \end{aligned} \tag{3}$$

SamplePairing, mixup_lp and extrapolation are sample mixed data augmentation of label preserving, which can used for Semi-supervised Learning.

4. EXPERIMENTS AND RESULTS

4.1. Dataset and preprocessing

The proposed methods are evaluated on CHIME-HOME dataset of the DCASE 2016 audio tagging challenge that comprises audio chunks along with corresponding multi-label annotations or ground truth labels [25]. The annotations are based on a set of 7 label classes, including child speech, adult male speech, adult female speech, video game / TV, percussive sounds, broadband noise and other identifiable sounds, denoted by c, m, f, v, p, b and o. Multi-label annotations suggest that the dataset is weakly labeled with chunk level labels rather than event level labels. In fact, an audio chunk may contain multiple sound events without indicating their occurrence time [6]. The development dataset consists of 1946 4-second chunks with the 16kHz sampling rate in mono. The target is to perform multi-label classification on 4-second audio chunks.
Directly learning features from the raw waveform is subject to the limited size of the training data. Presently, most of the audio tagging systems used frequency-domain features as the input, extracted from the audio signal clip. They are mainly borrowed from the field of speech recognition, such as mel-scale filter banks, log-frequency filter banks and time-frequency filters. In audio analysis tasks, for example, audio scene classification, audio event detection and audio tagging, frequency-domain representation provides superior performance. However, the MFCCs may not maintain locality by the discrete cosine transform projecting the spectral energies into a new basis. As a consequence, the log-mel features computed directly from the mel-frequency spectral coefficients for each frame of raw audio was used as an input of CNN [26], [27]. In this paper, we use log-mel features as the input for the neural network.

4.2. Evaluation and baseline

The official evaluation method for the challenge is average equal error rate (EER) for five-fold cross-validation. We followed the 5-fold cross-validation setting using the original folds splits. Early stopping is used to monitor the validation loss. Training is interrupted when the validation loss has not improved after 20 epochs. The batch size is set up to 44.
The EER is used as an evaluation metric, which is defined as the error rate at the ROC operating point where the false positive and false-negative rates are equal, and a lower EER represents better system performance.
We apply DAE-DNN [28], CGRNN [29] and ATT-LOC [21] as the baseline models. CGRNN and ATT-LOC downsamples the stereo audio data from the 48kHz sampling rate into 16kHz. Based on previous experiments, these three models achieved the state-of-the-art performance with 0.15,0.13 and 0.13 EER on the evaluation set, respectively.

4.3. Results

Table I shows experimental results of the DCASE 2016 audio tagging challenge, by using different data augmentation approaches. The experiment consists of three parts.

4.3.1. The effectiveness of the proposed architecture

Firstly, we verify the effectiveness of the proposed novel neural network architecture. without data augmentation, our proposed CRNN model (when $\alpha = 0$) can get acceptable classification performance with 0.13 EER. Incorporating mixup, when $\alpha = 1.5$ and $\alpha = 2$, gain the best performance with 0.10 EER, which is the state-of-the-art performance on the evaluation set of the DCASE 2016 audio tagging challenge.

4.3.2. Data augmentation of mixed form

we observe that data augmentation of mixed form (including mixup, SamplePairing and extrapolation) can effectively improve the classification. The only exception is the Samplepairing approach without fine tuning, no significant performance is observed. In more detail, SamplePairing without fine tuning perform poorly to classify the adult male speech (m) audio event. In the development dataset, the adult male speech (m) event occurs sparsely (number of occurrences is 174) with comparison with other events. New examples generated by fixed interpolation coefficient confused the classifier for minority classes.

Table 1: Experimental results on the evaluation set of the DCASE 2016 audio tagging challenge.

Model	Date augmentation	EER								Var(10^{-3})
		c	m	f	v	p	b	o	Avg	
DAE-DNN	$\sim$	0.21	0.15	0.21	0.02	0.18	0.01	0.26	0.15	9.45
CGRNN	IMD	0.17	0.16	0.18	0.03	0.15	0.00	0.24	0.13	7.39
ATT-LOC	IMD	0.09	0.14	0.17	0.03	0.12	0.01	0.24	0.11	6.36
CRNN	mixup($\alpha = 0.0$)	0.09	0.11	0.14	0.04	0.15	0.06	0.26	0.13	6.13
CRNN	mixup($\alpha = 0.1$)	0.10	0.23	0.15	0.02	0.15	0.03	0.23	0.13	7.30
CRNN	mixup($\alpha = 0.5$)	0.09	0.16	0.11	0.03	0.14	0.03	0.24	0.11	5.56
CRNN	mixup($\alpha = 1.0$)	0.09	0.12	0.11	0.02	0.12	0.03	0.26	0.11	6.25
CRNN	mixup($\alpha = 1.5$)	0.10	0.14	0.11	0.03	0.10	0.01	0.20	0.10	4.11
CRNN	mixup($\alpha = 2.0$)	0.10	0.11	0.11	0.03	0.11	0.00	0.25	0.10	6.28
CRNN	mixup($\alpha = 5.0$)	0.13	0.14	0.08	0.02	0.12	0.00	0.24	0.11	6.52
CRNN	SamplePairing	0.10	0.20	0.15	0.01	0.16	0.03	0.24	0.13	7.26
CRNN	mixup_lp($\alpha = 1.5$)	0.12	0.13	0.12	0.02	0.12	0.00	0.25	0.11	6.52
CRNN	extrapolation($\alpha = 1.5$)	0.10	0.16	0.13	0.03	0.14	0.02	0.23	0.12	5.43

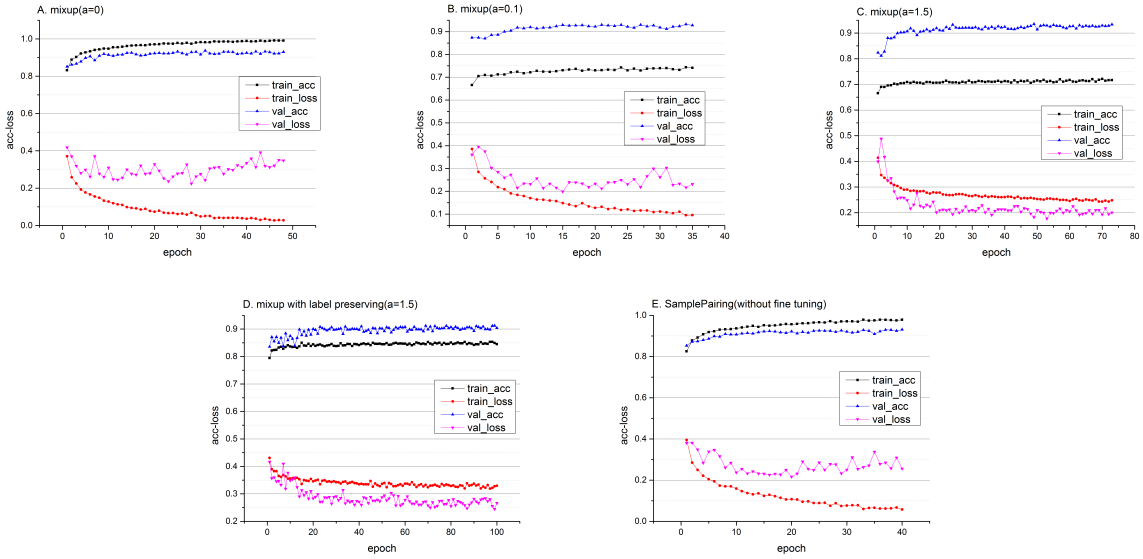


Figure 2: Model trainning curves using different data augmentation methods.

4.3.3. Different ratios for the mixup approach

we explored different ratios for the mixup approach. We choose $\alpha \in \{0, 0.1, 0.5, 1.0, 1.5, 2.0, 5.0\}$. We find that $\alpha \in \{1.5, 2.0\}$ gets the best performance in our experiment. In addition, when $\alpha = 1.5$, the approach has the smallest variance of EER 4.11×10^{-3}, which indicates that our CRNN model with mixup approach has better stability.

4.3.4. The model training history

Furthermore, Fig.2 shows the model training history for each epoch, including the training loss and accuracy, as well as the loss and accuracy for the validation dataset. As can be seen from the figure, the results in Fig.2B and Fig.2C get a significantly lower training accuracy (basically stable at around 0.7) and a higher training loss compared with that in Fig.2A (without mixup), whereas the validation loss is the lowest (when $\alpha = 1.5$). In addition, we observe that the bigger α, the lower training accuracy will be. When interpolation is not used for labels (as shown in Fig.2D), the training accuracy is obviously better than that in Fig.2C. However, this did not lead to performance improvements. In Fig.2A, Fig.2E and Fig.2F, the gap between training loss and validation loss increases as the training epoch increases. We suppose that this situation may be detrimental to the generalization ability of the model.

5. CONCLUSION

In this paper, for the audio tagging task, we use data augmentation of mixed form on the time-frequency representation in input space for training the convolutional recurrent neural network (CRNN). Experiments are conducted on DCASE 2016 (Task 4) task. Based on our experiments, sample-mixed based data augmentation can effectively improve the performance of audio tagging. Moreover, mixup generalizes better than other mixed form data augmentation methods, as it has dramatically decrease the gap between the training and test distribution, which gains the best performance with 0.10 EER, and this is the state-of-the-art performance on the evaluation set of the DCASE 2016 audio tagging challenge.

6. ACKNOWLEDGMENT

Qiuqiang Kong was supported by EPSRC grant EP/N014111/1 "Making Sense of Sounds" and a Research Scholarship from the China Scholarship Council (CSC) No. 201406150082.

7. REFERENCES

[1] S. Souli and Z. Lachiri, "Audio sounds classification using scattering features and support vectors machines for medical surveillance," *Applied Acoustics*, vol. 130, pp. 270–282, 2018.

[2] P. Cano, M. Koppenberger, and N. Wack, "Content-based music audio recommendation," in *ACM International Conference on Multimedia*, 2005, pp. 211–212.

[3] E. Sprengel, M. Jaggi, Y. Kilcher, and T. Hofmann, "Audio based bird species identification using deep learning techniques," 2016.

[4] http://www.cs.tut.fi/sgn/arg/dcase2016/challenge/.

[5] http://www.cs.tut.fi/sgn/arg/dcase2017/challenge/.

[6] A. Mesaros, T. Heittola, and T. Virtanen, "Tut database for acoustic scene classification and sound event detection," in *Signal Processing Conference (EUSIPCO), 2016 24th European*. IEEE, 2016, pp. 1128–1132.

[7] D. M. Hawkins, "The problem of overfitting," *Cheminform*, vol. 35, no. 19, p. 1, 2004.

[8] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[9] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv*, pp. 448–456, 2015.

[10] S. Park and N. Kwak, "Analysis on the dropout effect in convolutional neural networks," in *Asian Conference on Computer Vision*, 2016, pp. 189–204.

[11] X. Gastaldi, "Shake-shake regularization," *arXiv preprint arXiv:1705.07485*, 2017.

[12] Y. Yamada, M. Iwamura, and K. Kise, "Shakedrop regularization," *arXiv preprint arXiv:1802.02375*, 2018.

[13] P. Y. Simard, Y. A. Lecun, J. S. Denker, and B. Victorri, "Transformation invariance in pattern recognition tangent distance and tangent propagation," *International Journal of Imaging Systems and Technology*, vol. 11, no. 3, p. 181C197, 2000.

[14] J. Salamon and J. Bello, "Deep convolutional neural networks and data augmentation for environmental sound classification," *IEEE Signal Processing Letters*, vol. PP, no. 99, pp. 1–1, 2017.

[15] H. Inoue, "Data augmentation by pairing samples for images classification," 2018.

[16] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopezpaz, "mixup: Beyond empirical risk minimization," 2018.

[17] K. Choi, G. Fazekas, and M. Sandler, "Automatic tagging using deep convolutional neural networks," *arXiv preprint arXiv:1606.00298*, 2016.

[18] S. Adavanne and T. Virtanen, "Sound event detection using weakly labeled dataset with stacked convolutional and recurrent neural network," *arXiv preprint arXiv:1710.02998*, 2017.

[19] D. A. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (elus)," *Computer Science*, 2015.

[20] M. A. Nielsen, *Neural networks and deep learning*. Determination Press, 2015.

[21] Y. Xu, Q. Kong, Q. Huang, W. Wang, and M. D. Plumbley, "Attention and localization based on a deep convolutional recurrent model for weakly supervised audio tagging," *arXiv preprint arXiv:1703.06052*, 2017.

[22] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *Computer Science*, 2014.

[23] K. Xu, D. Feng, H. Mi, B. Zhu, D. Wang, L. Zhang, H. Cai, and S. Liu, "Mixup-based acoustic scene classification using multi-channel convolutional neural network," *arXiv preprint arXiv:1805.07319*, 2018.

[24] T. DeVries and G. W. Taylor, "Dataset augmentation in feature space," *arXiv preprint arXiv:1702.05538*, 2017.

[25] P. Foster, S. Sigtia, S. Krstulovic, J. Barker, and M. D. Plumbley, "Chime-home: A dataset for sound source recognition in a domestic environment," in *Applications of Signal Processing To Audio and Acoustics*, 2015, pp. 1–5.

[26] K. J. Piczak, "Environmental sound classification with convolutional neural networks," in *IEEE International Workshop on Machine Learning for Signal Processing*, 2015, pp. 1–6.

[27] O. Abdel-Hamid, A. R. Mohamed, H. Jiang, L. Deng, G. Penn, and D. Yu, "Convolutional neural networks for speech recognition," *IEEE/ACM Transactions on Audio Speech and Language Processing*, vol. 22, no. 10, pp. 1533–1545, 2014.

[28] Y. Xu, Q. Huang, W. Wang, P. Foster, S. Sigtia, P. J. Jackson, and M. D. Plumbley, "Unsupervised feature learning based on deep models for environmental audio tagging," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 25, no. 6, pp. 1230–1241, 2017.

[29] Y. Xu, Q. Kong, Q. Huang, W. Wang, and M. D. Plumbley, "Convolutional gated recurrent neural network incorporating spatial features for audio tagging," in *International Joint Conference on Neural Networks*, 2017.

MULTI-SCALE CONVOLUTIONAL RECURRENT NEURAL NETWORK WITH ENSEMBLE METHOD FOR WEAKLY LABELED SOUND EVENT DETECTION

Yingmei Guo, Mingxing Xu

Tsinghua University
Department of Computer Science and Technology, 30 ShuangQing Road
HaiDian, Beijing 100084, China
qniguoym@163.com

Jianming Wu, Yanan Wang, Keiichiro Hoashi

KDDI Research, Inc.
2-1-15 Ohara, Fujimino-shi, Saitama, 356-8502
Japan
{ji-wu, wa-yanan, hoashi}@kddi-research.jp

ABSTRACT

In this paper, we describe our contributions to the challenge of detection and classification of acoustic scenes and events 2018(DCASE2018). We propose multi-scale convolutional recurrent neural network(Multi-scale CRNN), a novel weakly-supervised learning framework for sound event detection. By integrating information from different time resolutions, the multi-scale method can capture both the fine-grained and coarse-grained features of sound events and model the temporal dependencies including fine-grained dependencies and long-term dependencies. CRNN using learnable gated linear units(GLUs) can also help to select the most related features corresponding to the audio labels. Furthermore, the ensemble method proposed in the paper can help to correct the frame-level prediction errors with classification results, as identifying the sound events occurred in the audio is much easier than providing the event time boundaries. The proposed method achieves 29.2% in the event-based F1-score and 1.40 in event-based error rate in development set of DCASE2018 task4 compared to the baseline of 14.1% F-value and 1.54 error rate[1].

Index Terms— Sound event detection, Weakly-supervised learning, Deep learning, Convolutional recurrent neural network, Multi-scale model

1. INTRODUCTION

Recently a large-scale weakly supervised sound event detection task of DCASE2018 challenge was proposed[2]. The target of the task is to provide not only the event class but also the event time boundaries given that multiple events can be present in an audio recording[3]. The task employs a subset of "Audioset: An Ontology And Human-Labeled Dataset For Audio Events" by Google[4].

Sound is one of the signals carrying information. The perception and understanding of sound plays an important role in human interaction with the surroundings. With the continuous development of smart homes, auto-driving cars and security surveillance devices, sound event detection has received increasing attention. With the development of multimedia and network technologies, audio data grows rapidly in the databases. Therefore, how to identify, label and retrieve useful content from audios effectively has become an urgent problem to be solved.

Many methods can be applied in the sound event detection task, such as Gaussian Mixture Models(GMM)[5], Hidden Markov Models(HMM)[6], non-negative matrix factorization(NMF)[7] and Deep Neural Network(DNN)[8]. ConvNets showed the promising results in a large number of computer vision tasks and have been actively adopted for audio content analysis[8]. [9] proposed a sound event detection system that combines global-input model and separated-input model using the entire and a segmented audio clip as input separately to predict audio events in a short-time segment. Moreover, convolutional neural network structures that perform well in image recognition tasks such as AlexNet[10], VGG[11], Inception[12] and ResNet[13] also perform well in sound event detection[14]. Eghbal-Zadeh et al. used the VGG classifier in the DCASE2016 acoustic scene recognition task and ranked first [15]. [16] proposed multi-scale RNN to balance the modeling of both the fine-grained and long-term dependencies.

As for the weakly labeled sound event detection, the weakly labeled data lacks frame-level strong labels. In [17], the frame-level prediction information was used as an intermediate variable, which can influence the final output of the model and be weakly supervised. [18] applied multi-instance learning(MIL) in sound event detection where each audio is regarded as a packet and frames or short segments are regarded as examples. [19] used the model proposed in [20] to do classification and applied transposed convolutional network to reconstruct the signal of original audio and make frame-level predictions. However, a significant amount of research is still needed to reliably detect sound events in realistic soundscapes, where multiple sounds are present, often simultaneously, and distorted by the environment.

In this paper, we propose multi-scale convolutional recurrent neural network. The CNN structure in the model is proposed by [17] which applies the learnable gated linear units(GLUs)[21] to replace the ReLU[22] activation after each layer of convolutional neural network. This learnable gate is able to control the information flow to the next layer. The RNN structure followed the CNN can model the temporal dependencies. The multi-scale method is applied to capture useful information from both the fine-grained and coarse-grained features of sound events and balance the modeling of both the fine-grained and long-term dependency. The ensemble method can further help to correct the frame-level prediction errors with classification results. Section 2 describes the our multi-scale CRNN architecture. Section 3 shows and discusses the experiments and results. In the end, section 4 summarizes and plans for the future work.

2. PROPOSED METHOD

2.1. Network Architecture

The main model structure is shown in Fig.1. The fine-grained input and the coarse-grained input of network are features described in the section 3.2 with the shape of (1,1200,64) and (1,240,64) separately. Some chunks extracted from audios with length shorter than 10 seconds are zero-padded to equalize the length.

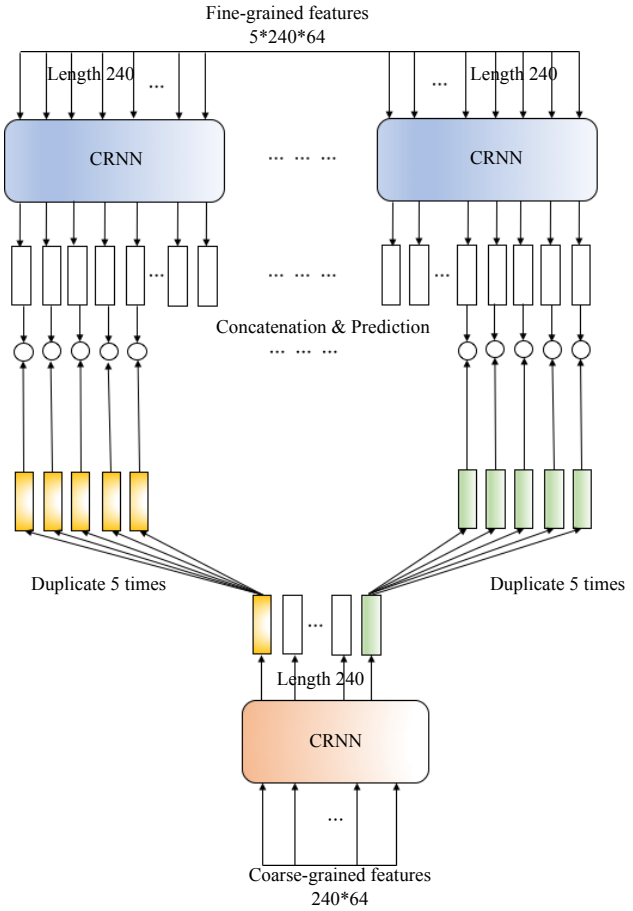


Figure 1: Multi-scale CRNN: Fine-grained feature sequence has a length of 1200 and coarse-grained feature sequence has a length of 240. During computation, the fine-grained sequence is splits into five subsequences to feed into the same CRNN structure with same parameters.

The CNN structure in the model is proposed by [17] which applies the learnable gated linear units(GLUs)[21] to replace the ReLU[22] activation after each layer of convolutional neural network. The motivation of using GLUs in audio classification is to introduce the attention mechanism to all layers of the neural network. GLUs are defined as:

$$Y = (W * X + a) \odot \sigma(V * X + b) \quad (1)$$

where X is the input of the first layer or the feature maps of the interval layers, W and V are the convolutional filters, a and b are the biases, σ is the sigmoid non-linearity, ⊙ is the element-wise product and * is the convolutional operator. It can attend to the T-F bin with related audio events by setting its value close to one otherwise close to zero and control the information passed on in the hierarchy. The RNN structure in the model is bidirectional to learn useful contextual information from both time directions. Fig.2 shows the CRNN structure.

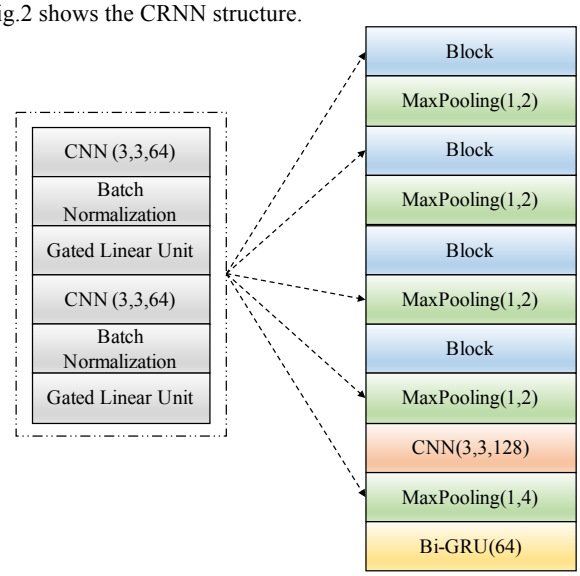


Figure 2: CRNN structure. The left part describes the details of the "Block".

The multi-scale method used in the model combines two CRNNs separately work at fine-scale and coarse-scale. The multi-scale CRNN can capture useful information from both the fine-grained and coarse-grained features of sound events and balance the modeling of both the fine-grained and long-term dependency. The last layers of multi-scale models are aligned so that each cell of the coarse-scale CRNN interacts with five cells of the fine-scale CRNN by concatenation.

After concatenation, final probabilistic predictions are made at fine-scale using a fully connected layer with sigmoid output:

$$y_t = \sigma\left(W\left(h1_t, h2_{\lceil \frac{t}{5} \rceil}\right) + b\right) \quad (2)$$

Where $h1_t, h2_{\lceil \frac{t}{5} \rceil}$ are the output of fine-scale CRNN at time t and coarse-scale CRNN at time $\lceil \frac{t}{5} \rceil$ separately. It should be noted that we only do convolution and pooling operations on spectral axis to keep the time resolution of the input.

As no frame-level strong labels are provided, the temporal information of each occurring sound event in the audio can only be weakly supervised inferred as intermediate variables. We aggregate probabilistic predictions of all frames to determine the existence of the event in that audio clip.

2.2. Ensemble Method

As the audio clips are weakly labeled, it is easy to do classification instead of detection. As a result of that, it is obvious the classification task can achieve higher accuracy. The ensemble method can help to correct the frame-level prediction errors with classification results.

We use the CNN based model introduced in [9] and a single-scale model similar with the model proposed in the paper to do classification and fuse the results of three models to produce the final predictions. Fig.3 shows the structure of the CNN based model.

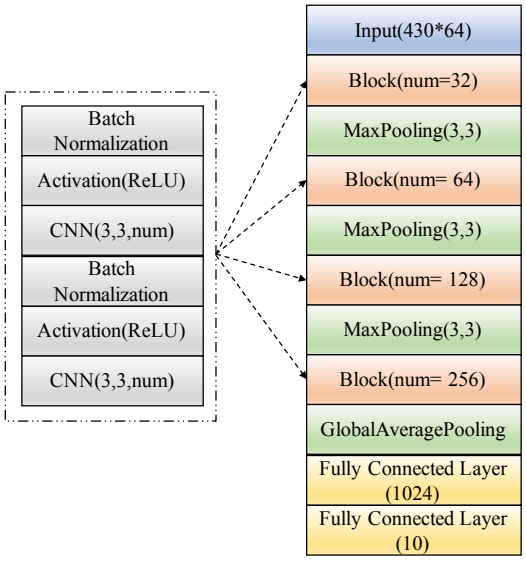


Figure 3: CNN based classification model architecture. The left part describes the details of the "Block" and the number in the brackets of the "Block" is the number of filters of convolutional layers.

We get final classification results using fusion method. The method is defined as:

$$R_i = \begin{cases} 1, r1_i + r2_i + r3_i \geq 2 \\ 0, r1_i + r2_i + r3_i < 2 \end{cases} \quad (3)$$

where R_i is the label of the i-th audio, $r1_i$, $r2_i$, $r3_i$ are the labels of the i-th audio with the single scale model, multi-scale model and CNN model separtely.

We use the classification models as sound event detectors and correct the frame-level prediction errors. When the event occurs in a frame it must occur in that audio clip.

3. EXPERIMENTS

3.1. Task and Data

The target of the DCASE2018 task4 is to provide not only the event class but also the event time boundaries given that multiple events can be present in an audio recording.

The data are Youtube videos excerpt from domestic context. We are focused on a subset of Audioset that consists of 10 classes of sound events: Speech, Dog, Cat, Alarm, Dishes, Frying, Blender, Running Water, Vacuum cleaner and Electric shaver.

3.2. Set-up

Log-Mel filter banks are used as our features. In general, we resample all audios to 44.1kHz and calculate the mel-spectrograms with 64 mel-bins at two scales with hop sizes of 0.0415 seconds and 0.0083 seconds, denoted as the coarse-scale and the fine-scale respectively. The window size of the two scales for short-time Fourier transform is 0.064 seconds. Then the resulting mel-spectrogram is converted into logarithmic scale and standardized by subtracting the mean value and dividing by the standard deviation. There are some audios that are shorter than 10 seconds, and the features extracted from the audios are zero-padded to equalize the length.

As shown in Fig.2, each convolutional network in the block has 64 filters with 3*3 size. The convolution network out of block has 128 filters with 3*3 size. The pooling size is 1*2 behind the block and 1*4 after the single convolution layer. One bidirectional gated recurrent neural network with 64 units is used.

In the training phase, we apply the binary cross-entropy loss between the predicted probability and ground truth of an audio recording. Adam[23] is used as the stochastic optimization method.

3.3. Results

The results of audio tagging and weakly supervised sound event detection will be given in this section.

3.3.1. Audio tagging

Table 1 shows the Precision, Recall and F1-value of multiple different systems on development set of DCASE2018 task4. We can find that multi-scale CRNN model is better than single scale CRNN with F1-value of 85.5%. We also fuse the three models by soft voting. The fusion model achieves the best score.

Table 1. Comparison of multi-scale CRNN, single-scale CRNN, CNN based model and fusion model on development set of DCASE2018 task4.

Models	Precision(%)	Recall(%)	F1-value(%)
CNN based model[9]	85.1	85.1	85.1
Single-scale CRNN[17]	83.2	80.9	82.0
Multi-scale CRNN	83.5	87.6	85.5
Fusion	87.7	89.8	88.7

3.2.2 Sound event detection

Table 2 shows the F1-value and error rate of multi-scale CRNN using and not using classification results for correction. It show that post-processing of the frame-level predictions is important and can improve the system performance by 6.1%. Submissions are evaluated with event-based measures with a 200ms collar on onsets and a 200ms / 20% of the events length collar on offsets.

Table 2. Comparison of F1-value and the error rate of multi-scale CRNN using and not using classification results for correction on development set of DCASE2018 task4.

Models	F1-value(%)	Error rate
Multi-scale CRNN not using correction	23.1	1.90
Multi-scale CRNN using correction	29.2	1.40

Table 3 shows the F1-value and the error rate of single-scale CRNN, multi-scale CRNN and the baseline on development set of DCASE2018 task4. Multi-scale CRNN has the best performance. It demonstrates that multi-scale method can capture useful

informational from both the fine-grained and coarse-grained features of sound events and balance the modeling of both the fine-grained and long-term dependency.

Table 3. Comparison of multi-scale CRNN, single-scale CRNN, and baseline on development set of DCASE2018 task4.

Models	F1-value(%)	Error rate
Baseline[1]	14.1	1.54
Single-scale CRNN[17]	24.4	1.24
Multi-scale CRNN	29.2	1.40

4. CONCLUSIONS

In this paper, we propose multi-scale convolutional re-current neural network. The CNN structure in the model applies the learnable gated linear units to control the information flow to the next layer. The RNN structure followed the CNN can model the temporal dependencies. The multi-scale method is applied to capture useful information from both the fine-grained and coarse-grained features of sound events. It also balances the modeling of both the fine-grained and long-term dependency. The ensemble method can help to correct the frame-level prediction errors with classification results.

We also tried several methods to improve the system using unlabeled data but we are not satisfied with the results achieved. To further improve the system, future work can be done by exploring the possibility to exploit a large amount of unlabeled and unbalanced training data together with a small weakly annotated training set.

5. REFERENCES

[1] R. Serizel, N. Turpault, H. Eghbal-Zadeh, A. P. Shah . “Large-Scale Weakly Labeled Semi-Supervised Sound Event Detection in Domestic Environments ”. Submitted to DCASE2018 Workshop, 2018.

[2] Romain Serizel, Nicolas Turpault, Hamid Eghbal-Zadeh, and Ankit Parag Shah. Large-Scale Weakly Labeled Semi-Supervised Sound Event Detection in Domestic Environments. Submitted to DCASE2018 Workshop, July 2018.

[3] Kumar, A., & Raj, B. (2016, October). Audio event detection using weakly labeled data. In Proceedings of the 2016 ACM on Multimedia Conference (pp. 1038-1047). ACM.

[4] Gemmeke, J. F., Ellis, D. P., Freedman, D., Jansen, A., Lawrence, W., Moore, R. C., ... & Ritter, M. (2017, March). Audio set: An ontology and human-labeled dataset for audio events. In Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on (pp. 776-780). IEEE.

[5] Grégoire Lafay, Mathieu Lagrange, Mathias Rossignol, Emmanouil Benetos, Axel Roebel, "A Morphological Model for Simulating Acoustic Scenes and Its Application to Sound Event Detection", Audio Speech and Language Processing IEEE/ACM Transactions on, vol. 24, no. 10, pp. 1854-1864, 2016.

[6] Heittola, T., Mesaros, A., Eronen, A., & Virtanen, T. (2013). Context-dependent sound event detection. EURASIP Journal on Audio, Speech, and Music Processing, 2013(1), 1.

[7] Dikmen, O., & Mesaros, A. (2013, October). Sound event detection using non-negative dictionaries learned from annotated overlapping events. In Applications of Signal Processing to Audio and Acoustics (WASPAA), 2013 IEEE Workshop on (pp. 1-4). IEEE.

[8] Su T W, Liu J Y, Yang Y H. Weakly-supervised audio event detection using event-specific Gaussian filters and fully convolutional networks[C]// IEEE International Conference on Acoustics, Speech and Signal Processing. IEEE, 2017:791-795.

[9] Lee D, Lee S, Han Y, et al. ENSEMBLE OF CONVOLUTIONAL NEURAL NETWORKS FOR WEAKLY-SUPERVISED SOUND EVENT DETECTION USING MULTIPLE SCALE INPUT[C]// DCASE 2017 Workshop. 2017.

[10] Krizhevsky A, Sutskever I, Hinton G E. ImageNet classification with deep convolutional neural networks[C]// International Conference on Neural Information Processing Systems. Curran Associates Inc. 2012:1097-1105.

[11] Su, T. W., Liu, J. Y., & Yang, Y. H. (2017, March). Weakly-supervised audio event detection using event-specific gaussian filters and fully convolutional networks. In 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (pp. 791-795).

[12] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 2818-2826).

[13] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).

[14] Hershey, S., Chaudhuri, S., Ellis, D. P., Gemmeke, J. F., Jansen, A., Moore, R. C., ... & Slaney, M. (2017, March). CNN architectures for large-scale audio classification. In Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on (pp. 131-135). IEEE.

[15] Eghbal-Zadeh, H., Lehner, B., Dorfer, M., & Widmer, G. (2016). CP-JKU submissions for DCASE-2016: A hybrid approach using binaural i-vectors and deep convolutional neural networks. IEEE AASP Challenge on Detection and Classification of Acoustic Scenes and Events (DCASE).

[16] Rui Lu, Zhiyao Duan, Changshui Zhang. Multi-scale recurrent neural network for sound event detection[R]. Technical report, DCASE2017 Challenge (September 2017), 2017.

[17] Xu, Y., Kong, Q., Wang, W., & Plumbley, M. D. (2017). Large-scale weakly supervised audio classification using gated convolutional neural network. arXiv preprint arXiv:1710.00343

[18] Salamon J, McFee B, Li P, et al. DCASE 2017 submission: Multiple instance learning for sound event detection[R]. Technical report, DCASE2017 Challenge (September 2017), 2017.

[19] Chou, S. Y., Jang, J. S. R., & Yang, Y. H. (2017). FrameCNN: a weakly-supervised learning framework for frame-wise acoustic event detection and classification. Recall, 14, 55-4.

[20] Van den Oord, A., Dieleman, S., & Schrauwen, B. (2013). Deep content-based music recommendation. In Advances in neural information processing systems (pp. 2643-2651).

[21] Hershey, S., Chaudhuri, S., Ellis, D. P., Gemmeke, J. F., Jansen, A., Moore, R. C., ... & Slaney, M. (2017, March). CNN architectures for large-scale audio classification. In Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on (pp. 131-135). IEEE.

[22] Eghbal-Zadeh, H., Lehner, B., Dorfer, M., & Widmer, G. (2016). CP-JKU submissions for DCASE-2016: A hybrid approach using binaural i-vectors and deep convolutional neural networks. IEEE AASP Challenge on Detection and Classification of Acoustic Scenes and Events (DCASE).

[23] Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

EXPLORING DEEP VISION MODELS FOR ACOUSTIC SCENE CLASSIFICATION

Octave Mariotti, Matthieu Cord, Olivier Schwander

Sorbonne Université, CNRS
Laboratoire d'Informatique de Paris 6, LIP6
F-75005 Paris, France
name.surname@lip6.fr

ABSTRACT

This report evaluates the application of deep vision models, namely VGG and Resnet, to general audio recognition. In the context of the IEEE AASP Challenge: Detection and Classification of Acoustic Scenes and Events 2018, we trained several of these architectures on the task 1 dataset to perform acoustic scene classification. Then, in order to produce more robust predictions, we explored two ensemble methods to aggregate the different model outputs. Our results show a final accuracy of 79% on the development dataset for subtask A, outperforming the baseline by almost 20%.

Index Terms— Acoustic scene classification, DCASE 2018, Vision, VGG, Residual networks, Ensemble

1. INTRODUCTION

Despite remarkable improvement over the last decade in computer vision, its neighbor field audio signal processing seems to be lagging behind, a delay often attributed to the lack of available labeled data to train deep networks. Nonetheless, recent results show that if sufficient data is provided, vision models seem to perform quite well in audio classification tasks[1], implying that a transition towards deeper models will soon occur.

Current state-of-the-art algorithm for acoustic scene classification rely on preprocessing audio signal to obtain a spectral representation of the signal, and then training a convolutional neural network to classify the data. There does not seem to be a general consensus on which features are best, although most commonly found are mel-spectrograms[2] and Mel Frequency Cepstral Coefficients (MFCC)[3]. Other popular features include standard or Constant-Q Transforms spectrograms[4] or more complex constructions such as i-vectors [5]. It should be noted that apart from regular and CQT spectrograms, all other preprocessing are specifically designed for music or speech processing, thus may introduce bias in a general audio dataset.

Classification of these features is almost always performed using CNNs whose architecture are loosely based on VGG networks [2][4]. Variation exists, often including recurrent layer to capture the temporal structure of audio signal[6], and Support Vector Machines are sometimes used to better classify high-level extracted features[4]. Models rarely exceed ten layers.

In comparison, current vision models are often deeper because image datasets contain much more examples. Nonetheless, some network architectures, although deep, are remarkably strong at avoiding overfitting, using regularization layers such as dropout or batch normalization. We intend to explore how these network behave on tasks they were not directly designed for.

Ensemble methods are often used to obtain better predictions either via direct methods like voting[6] or indirect, such as fitting an auxiliary classifier[4].

Our contribution (figure 1) relies on testing deep models with respect to the current status of general audio processing. Different depths and structures are evaluated in order to determine up to how many layers can a model grow before overfitting, and later merged with ensemble methods to benefit from different levels of abstraction.

2. PREPROCESSING

The preprocessing of audio sample is based on the COCAI DCASE 2017 submission[2], using several kind of log-mel spectrograms. The mel-spectrograms were computed from audio file converted from 24 to 16-bit encoding, with original 48kHz sample rate. The window was 2,048 samples long with a hop length of 1,024, for 128 mel bins. After converting the spectrograms to logarithmic scale, a final step of normalization was performed, consisting in subtracting μ and dividing each spectrogram by σ, where μ and σ are the average mean and standard deviation of spectrograms obtained from the development dataset.

Four types of spectrograms were created and used to train networks: Mono, Stereo, Mid / Side and Harmonic / Percussive. The mono preprocessing produces one mel-spectrogram per audio file, while the remaining three types produce a pair.

A stereo pair consists in a spectrogram extracted from the

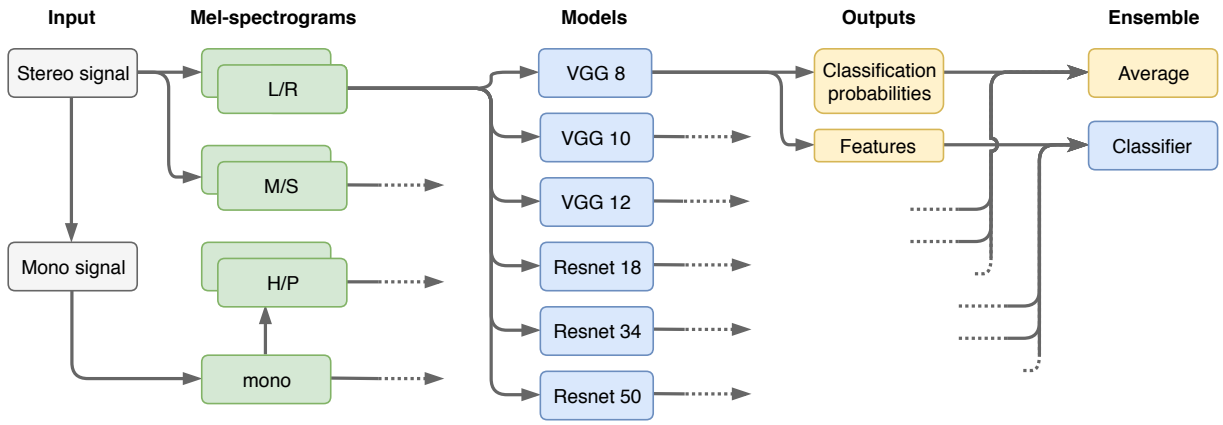


Figure 1: Architecture of the whole system

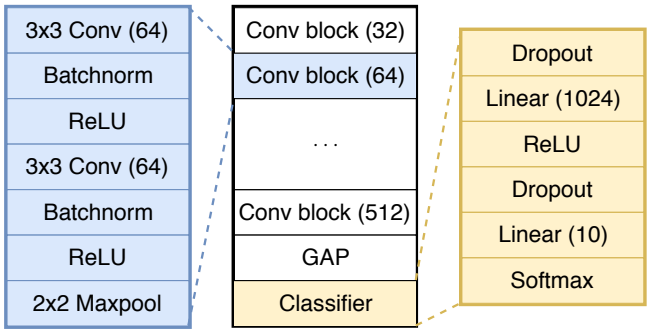


Figure 2: Architecture of VGG-inspired network

left (L) and right (R) channel respectively, and a mid/side pair from $L + R$ and $L - R$ channels.

Harmonic / Percussive preprocessing also produces two spectrograms, one containing harmonic features (or stationary frequencies) the other containing percussive features (or transient frequencies). Harmonic Percussive Source Separation (HPSS) first originated in music processing to isolate other instruments from the drum sounds that often made signals harder to process[7]. The algorithm used to perform HPSS was the median-based algorithm[8] from librosa audio processing library[9] with default parameters.

3. NETWORK ARCHITECTURES

We explored two kinds of network architectures inspired by well-tested vision models : VGG and residual networks.

3.1. VGG

VGG networks[10] have a straightforward architecture, consisting in a sequence of blocks containing two or three convolution layers followed by max-pooling. Because of the pooling layers, the spatial extent of the signal is reduced after each block. Conversely, the number of channels increases, allowing for more abstraction. We chose to limit the maximum number of channels to 512. After several of these blocks, two fully-connected layers are used to classify features.

Our architecture uses the same principle with two distinct features : regularization layer and global average pooling (figure 2).

Regularization layers Between each convolution layer and its activation function is inserted a batch normalization layer[11]. This introduces very few parameters and greatly improves generalization, yielding up to 10% accuracy improvement. Additionally, the classifier also incorporated dropout layer before fully-connected filter[12].

Global average pooling Instead of flattening the features maps obtained at the end of the convolution sequence and plugging the MLP on the resulting features, we chose to use global average pooling, averaging each feature map both on the time and frequency domain. This reduces the number of parameters with no apparent loss on accuracy. Moreover, GAP layers have been extensively used in object localization tasks[13], making this architecture interesting for audio segmentation.

We used three models totalizing 8, 10 and 12 layers respectively, including the fully-connected layers. Larger sizes were shown to overfit. These models are much shallower than actual VGG networks used in computer vision (at least 16 layers), but the drastically reduced number of parameters (5.3 versus 138 millions) allows them to cope with the small amount of examples in the dataset.

Figure 3: Architecture of a residual block

3.2. Residual networks

Residual networks[14] have introduced the concept of skip-connection, that have been reused in many architectures, including audio processing tasks[15]. It consists in adding the input to the output of a convolution block, with the initial motivation of not harming performances when adding layers (figure 3). These skip connections coupled with batch normalization make them remarkably robust to overfitting.

We have tested several standard sizes of residual networks in order to assess their behavior on a relatively small dataset, selecting three in the end: 18, 34 and 50. These are canonical architectures described in the original paper, the only addition we made was a dropout layer between the GAP layer and the fully-connected classifier. Note that the 50-layered version uses bottlenecks blocks while the other two use standard ones.

3.3. Training

Our networks were trained using the data segmentation provided with the data, with 15% of training data reserved for validation, on each of the four dataset of mel-spectrograms. When processing a pair of spectrograms (L/R, M/S and H/P), we set the input channels of our networks to two. Training was performed using gradient-descent with momentum, with a batch-size of 64, an initial learning rate of 10^{-2} divided by 2 if no improvement was observed on validation loss for 10 epochs. Because deeper architecture are more likely to overfit, we carefully selected training hyperparameters to obtain the best test accuracy. In particular, we stopped the training if the average loss did not improve over a sliding window spanning the last 30 epochs.

After training, each network produced a set of predictions and a set of features for each dataset. The features are defined as the output of the GAP layer for both architectures. These outputs were stored in order to perform ensemble methods (figure 1).

4. ENSEMBLING

Ensemble methods are a common technique used to improve generalization, involving making a classification choice

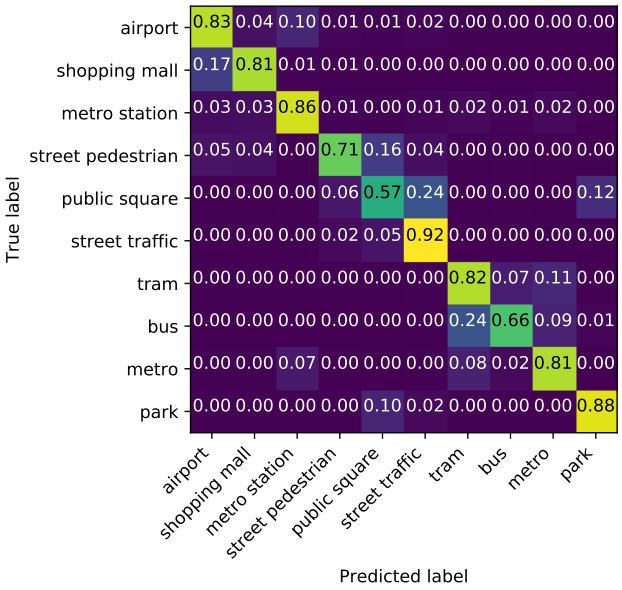


Figure 4: Confusion matrix of the NN no50 system

based on the output of several models. Common ensemble methods include voting, taking the maximum average classification probability over models or using special algorithms designed to optimize the weight of each model.

We explored two methods : mean of probabilities and training a multi-layer perceptron on extracted features.

Mean probabilities Averaging prediction probabilities is a straightforward method that provides results a bit more robust compared to voting, as it is more stable on difficult examples because outputs are not binary.

MLP We hoped that a multi-layer perceptron trained on the extracted features can detect complex patterns across models and perform better than simple ensembling of outputs. However, it is also prone to overfitting, as extracted feature are already very expressive. Consequently, we tried to train it with different subsets of extracted features.

In the end, each set of spectrograms was used to train 6 models, so the ensemble was performed on up to 24 models in total.

5. EXPERIMENTAL RESULTS

The experimental results in table 1 tend to show that VGG-like models perform better, most likely because residual networks tend to overfit the data. Nonetheless, they exhibit interesting performances, with the 50-layer version showing little accuracy drop (2-3%) compared to the 18-layer one despite having more than twice as many layers.

Model	Mono	L/R	M/S	H/P
VGG8	67.9	68.4	73.8	68.5
VGG10	72.8	70.9	77.7	72.8
VGG12	74.6	72.5	77.1	76.2
Resnet18	71.0	70.8	73.7	72.2
Resnet34	69.1	65.6	72.3	68.1
Resnet50	69.2	68.9	71.0	69.4

Table 1: Test accuracy of each network

Ensemble	Decision method	Resnet50	Accuracy
MP all	Mean probabilities	included	78.4
MP no50	Mean probabilities	excluded	79.1
NN all	Neural network	included	76.4
NN no50	Neural network	excluded	79.3

Table 2: Accuracy of ensembles

Mid / Side preprocessing consistently outperforms its peers, most likely because the side channel isolates interesting spatial information of the signal.

Both ensemble techniques were tried on two subset of models : the whole set, or all VGG models and Resnets 18 and 34. This is because Resnet 50 results show severe signs of overfitting, thus might hinder predictions.

The mean probabilities ensemble allowed to reach an accuracy of 79.1, a small improvement the best model (VGG10 M/S). Although the gain was only marginal, we expect that generalization was improved.

The trained MLP achieved a final accuracy of 79.3 when trained on all features but those obtained from Resnet50. We tried isolating some training example to use them only for the training of the aggregating classifier, but this yielded poor results as either extracted features were not good enough, or the classifier overfitted the data. Although VGG8 display similar accuracies, it is most likely caused by underfit, thus less problematic when ensembling.

The confusion matrix (figure 4) shows problematic confusion between several classes: airport and shopping mall, tram and bus, and most surprisingly, public square and street traffic.

5.1. Other experiments

Other architectures were also considered but dropped because they performed poorly:

Attention Both VGG-like and Resnet architectures were tested with the addition of gated activation function, often referred to as attention mechanism. Such designs were first introduced by vision models[16] and popularized on audio processing models by Wavenet[15]. However, experiments did not show a statistically significant improvement from this design compared to simply fitting a ReLU activation function.

Raw audio Models were also adapted to work on raw audio, effectively by swapping all 2D convolution filters with 1D equivalents. First results were promising, but these models were difficult to train, because of the time and memory needed to run them, and hyperparameter optimization was fairly complex. We have also tested two existing architectures working with raw audio - Envnet[17] and Wavenet[15] - but their results were only slightly above the baseline. Moreover, because these models work with lower-level data, they need more layers compared to spectral-based models, resulting in higher overfit probability.

Cepstral features Models were also trained using cepstral features, namely MFCC obtained from the mel spectrograms, but this yielded a significant drop in accuracy, so they were not considered for the ensemble methods.

Transfer In order to further test our models, we tried to prevent overfitting with transfer learning. Using features extracted from image datasets could indeed provide a form of regularization[18]. Both a simple transfer strategy and a transfer with finetunig of weights have been tried on the Resnet architecture , but none were able to match the performances of end-to-end models.

6. CONCLUSION

Our system using an ensemble of classical vision models to classify acoustic scenes yielded an improvement of nearly 20 points in accuracy compared to the baseline of the challenge. Although not really original, our approach allowed to highlight the importance of regularization layers, in particular batch normalization, and validated the use of global average pooling layers to reduce number of parameters.

Despite residual networks not being completely sheltered from overfitting, they proved to perform well with respect to their depth : our biggest residual network had five times as many layers as our VGG architectures, but still managed to learn reasonably. As such we believe very deep models - especially featuring skip connections - might gain popularity in audio recognition once larger datasets are available.

Another possible system could be trained end-to-end with each network outputting features and a global classifier trained on the aggregation of the networks outputs, allowing each to specialize on different aspects of the signal. However, this requires holding many parameters in memory thus severely limits the size of networks used.

7. REFERENCES

[1] S. Hershey, S. Chaudhuri, D. P. Ellis, J. F. Gemmeke, A. Jansen, R. C. Moore, M. Plakal, D. Platt, R. A. Saurous, B. Seybold, *et al.*, "Cnn architectures for large-scale audio classification," in *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*. IEEE, 2017, pp. 131–135.

[2] Y. Han and J. Park, "Convolutional neural networks with binaural representations and background subtraction for acoustic scene classification," DCASE2017 Challenge, Tech. Rep., September 2017.

[3] R. Hyder, S. Ghaffarzadegan, Z. Feng, and T. Hasan, "BUET bosch consortium (B2C) acoustic scene classification systems for DCASE 2017," DCASE2017 Challenge, Tech. Rep., September 2017.

[4] Z. Weiping, Y. Jiantao, X. Xiaotao, L. Xiangtao, and P. Shaohu, "Acoustic scene classification using deep convolutional neural network and multiple spectrograms fusion," DCASE2017 Challenge, Tech. Rep., September 2017.

[5] B. Lehner, H. Eghbal-Zadeh, M. Dorfer, F. Korzeniowski, K. Koutini, and G. Widmer, "Classifying short acoustic scenes with I-vectors and CNNs: Challenges and optimisations for the 2017 DCASE ASC task," DCASE2017 Challenge, Tech. Rep., September 2017.

[6] S. Mun, S. Park, D. Han, and H. Ko, "Generative adversarial network based acoustic scene training set augmentation and selection using SVM hyper-plane," DCASE2017 Challenge, Tech. Rep., September 2017.

[7] N. Ono, K. Miyamoto, J. Le Roux, H. Kameoka, and S. Sagayama, "Separation of a monaural audio signal into harmonic/percussive components by complementary diffusion on spectrogram," in *Signal Processing Conference, 2008 16th European*. IEEE, 2008, pp. 1–4.

[8] D. Fitzgerald, "Harmonic/percussive separation using median filtering," in *13th International Conference on Digital Audio Effects (DAFx-10)*, 2010.

[9] B. McFee, C. Raffel, D. Liang, D. P. Ellis, M. McVicar, E. Battenberg, and O. Nieto, "librosa: Audio and music signal analysis in python," in *Proceedings of the 14th python in science conference*, 2015, pp. 18–25.

[10] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[11] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[12] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[13] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, "Learning deep features for discriminative localization," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2921–2929.

[14] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[15] A. Van Den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. W. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio." in *SSW*, 2016, p. 125.

[16] A. van den Oord, N. Kalchbrenner, L. Espeholt, O. Vinyals, A. Graves, *et al.*, "Conditional image generation with pixelcnn decoders," in *Advances in Neural Information Processing Systems*, 2016, pp. 4790–4798.

[17] Y. Tokozume and T. Harada, "Learning environmental sounds with end-to-end convolutional neural network," in *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*. IEEE, 2017, pp. 2721–2725.

[18] H. Bahuleyan, "Music genre classification using machine learning techniques," *arXiv preprint arXiv:1804.01149*, 2018.

3D CONVOLUTIONAL RECURRENT NEURAL NETWORKS FOR BIRD SOUND DETECTION

Ivan Himawan, Michael Towsey, Paul Roe

Science and Engineering Faculty, Queensland University of Technology
Brisbane, Australia
{i.himawan, m.towsey, p.roe}@qut.edu.au

ABSTRACT

With the increasing use of a high quality acoustic device to monitor wildlife population, it has become imperative to develop techniques for analyzing animals' calls automatically. Bird sound detection is one example of a long-term monitoring project where data are collected in continuous periods, often cover multiple sites at the same time. Inspired by the success of deep learning approaches in various audio classification tasks, this paper first reviews previous works exploiting deep learning for bird audio detection, and then proposes a novel 3-dimensional (3D) convolutional and recurrent neural networks. We propose 3D convolutions for extracting long-term and short-term information in frequency simultaneously. In order to leverage powerful and compact features of 3D convolution, we employ separate recurrent neural networks (RNN), acting on each filter of the last convolutional layers rather than stacking the feature maps in the typical combined convolution and recurrent architectures. Our best model achieved a preview of 88.70% Area Under ROC Curve (AUC) score on the unseen evaluation data in the second edition of bird audio detection challenge. Further improvement with model adaptation led to a 89.58% AUC score.

Index Terms— bird sound detection, deep learning, 3D CNN, GRU, biodiversity

1. INTRODUCTION

There has been growing interest to assess the wide-ranging impacts on biodiversity currently occurring around the globe. With the rapid decline in global wildlife populations due to environmental pollution, there has been a progressive effort over the years for monitoring vocalizing species as valid indicators of biodiversity. Monitoring the avian population in their habitats is one such effort since birds are good ecological indicators of environmental changes [1]. For example, this enables researchers to obtain valuable information such as habitat change, migration pattern, pollution, and disease outbreaks in the environments. Because birds play a crucial role in the environment, considerable effort has been devoted to focusing on the conservation of birds.

In order to collect data on a large spatio-temporal scale, ecologists often deploy acoustic monitoring devices to cover a large area of the land. As a result, a large number of recordings are being generated. These recordings, constituting many years of environmental monitoring, cannot be analyzed manually. In this regard, ecoacoustics research [2, 3] has become one of the "big data" research areas and may benefit substantially from "big data" analysis. Detecting bird sounds in audio recordings is one research problem example where data are continuously collected from various sources in a wide range of locations and environments, including from mobile phones [4]. This task can be extremely difficult to deal with due to man-made noise (i.e., traffic, television) , weather noise (e.g., rain, wind) , non-bird calls, and the quality of recordings.

In recent years, deep learning techniques have revolutionized the applicability of machine learning in speech, vision, and text processing. Significant improvements in many classification tasks are reported using deep architectures, where deep convolutional neural networks (CNN) have been used extensively in computer vision tasks. Since CNN learn filters that are shifted in both frequency and time, it addresses the limitation of deep neural networks (DNN), which lacks both time and frequency invariance. The use of deeper and more efficient CNN (e.g., GoogLeNet, ResNet, DenseNet) is also becoming popular and has shown state-of-the-art performance in object detection and image classification challenges [5, 6, 7]. The use of CNN is also popular in audio classification and speech recognition applications where audio signal is often converted into a spectrogram and treated as an input image to CNN. Despite this, bird sound data still pose a challenging problem for a deep learning method. This is not only due to environmental noise but also the complex structure and temporal modulations of bird songs [8, 9].

Our novel contribution in this paper is the extension of conventional convolutional recurrent neural networks using 3-dimensional (3D) convolutional architecture for bird sound detection. The 3D CNN architecture has been employed in video processing applications such as human action classification [10], audio-visual matching [11], and recently text-independent speaker verification [12]. In this work, we use 3D CNN to capture both long-term and short-term information in frequency from audio data stream. Also, 3D CNN is assumed to produce powerful and compact features compared to 2D CNN [13]. In order to receive the greatest benefit from these features, we employ separate RNN, acting on each filter of the last convolutional layers rather than stacking the feature maps in the typical combined CNN and RNN architectures.

The rest of the paper is organized as follows. Section 2 discusses related work. Section 3 describes data and methods for bird sound detection. Experimental results are presented and discussed in Section 4 and 5, respectively. Finally, Section 6 concludes the paper.

2. RELATED WORKS

Currently, the state-of-the-art results for bird sound detection, and also recognition of birds are obtained with the use of CNN. Specifically, CNN can act as a feature extractor which is shown to be superior to hand-crafted features in many classification tasks [14]. Thus, a mid-level representation of audio (i.e., a spectrogram) is popular as an input feature since it contains high-dimensional infor-

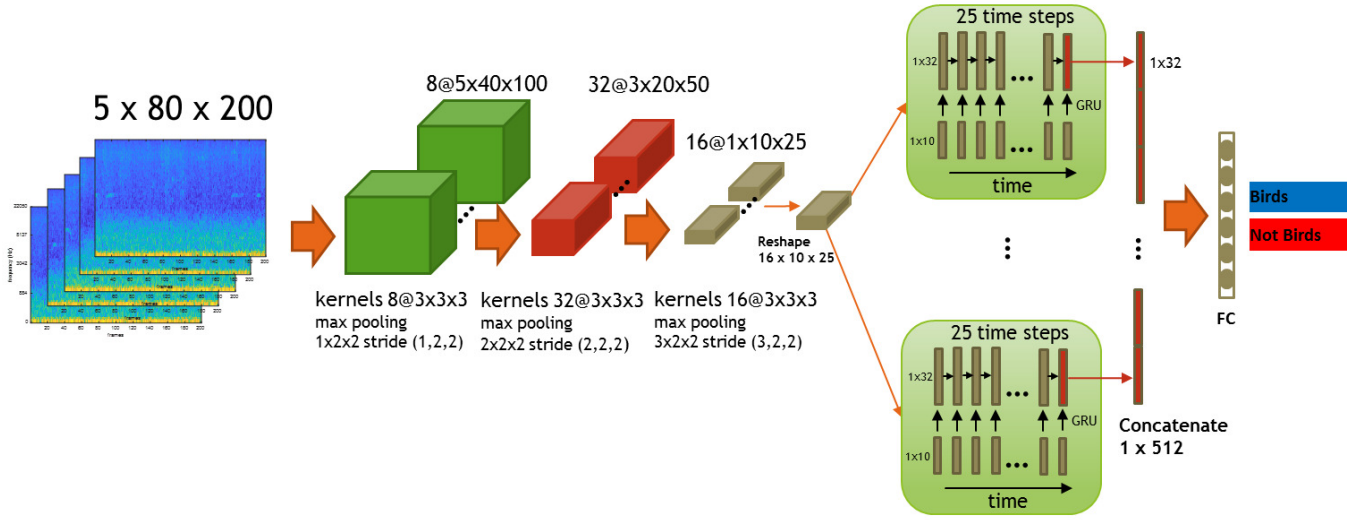


Figure 1: 3D-CNN architecture for bird sound detection. A 3D convolutional neural networks with three convolutional layers followed six teen recurrent layers and at the end one fully connected (FC) layer followed by softmax output layer. Input is a stack of 2-second audio clip.

mation (e.g., channel, environment). Despite promising detection results when using sophisticated classifiers such as CNN, state-of-the-art results can only be obtained if CNN is tuned carefully. This often requires domain knowledge and the interpretation of models that are well suited to bird data. The typical workflow for large scale bird sound detection and recognition using CNN consists of spectrogram feature extraction from audio recordings, and model training and evaluation. There is a considerable amount of work involved in predicting the location of bird sound within the spectrogram. The aim is to remove background noise and extract only the parts containing bird singing/calling [15]. This includes a spectral enhancement stage and image processing heuristics to discard non-bird sounds [16]. Even though noise reduction techniques may work well for certain datasets, bird sound localization is still a challenging task when there are dominant man-made noises (e.g., traffic, human singing, vehicles) in the audio clip.

A variety of CNN architectures have been explored for bird audio detection and recognition tasks. Very deep CNN networks such as ResNet [6] and DenseNet [7] architectures typically achieve better performance compared to the standard CNN model [17]. However, as shown in the previous BAD challenge, using a wide receptive field in a conventional CNN configuration can also achieved state-of-the-art results (*bulbul* submission). Other notable deep learning architecture employed in BAD challenge is the combination of CNN and RNN architectures (CNN+RNN) [18, 19]. In this case, the CNN is used for local feature extraction and the recurrent layers to model the long-term dependencies. For example, [19] used bi-directional RNN (BRNN) to process feature maps of the last CNN layer and achieved 88.41% AUC measure on the evaluation data. Data augmentation strategy (i.e., frequency and time shift) to improve the generalization of the network is also employed by many teams, albeit with marginal improvement [18]. We also tested our proposed 3D-CNN+RNN in the previous BAD evaluation set (post-challenge submission) and achieved 88.95% AUC score (without data augmentation method), comparable to the official state-of-the-art results published in the first challenge.

3. DATA AND METHODS

3.1. Datasets

Table 1: Bird audio detection challenge 2 statistics in the development set [20].

Dataset	present	absent	total
freefield1010	1,935	5,755	7,690
warblrb10k	6,045	1,955	8,000
BirdVox	10,017	9,983	20,000
Total	17,997	17,693	35,690

The bird audio detection challenge 2 used datasets released in previous challenge with the addition of new datasets: (a) BirdVox (BirdVox-DCASE-20k), and (b) Poland (PolandNFC), used only for evaluation. Each audio clip is 10-second long and sampled at 44.1 kHz. The total number of audio recordings for development and evaluation set are 35,690 and 12,620, respectively. The label for development set is 1 if any bird sound is present, regardless of the species, and 0 if none. The statistics of the development sets are presented in Table 1.

3.2. Feature Extraction

We split 10-second audio clip into $5 \times$ 2-second clips. The 2-second length is based on empirical analysis [21]. A spectrogram (from 2-second clip) computed from sequences of Short-Time Fourier Transform (STFT) of overlapping windowed signals is used as the sound representation. A signal is framed using a window of 20 ms (882 samples). The STFT analysis is carried out using a Hamming window, 50% overlap, 1024 FFT bins by zero padding. Given the audio signal $s(t)$, the square of magnitude spectrum $|S(n, f)|$ at frame n and frequency f is computed. We constructed triangular-shape filters linearly spaced in mel scale to convert a spectrogram to

a Mel-spectrogram with the number of filters set to 80. The magnitude values are then converted into log magnitude. The input feature shape for spectrogram is $5 \times 80 \times 200$. The features were standardized as input to 3D-CNN.

3.3. 3D convolutional recurrent neural networks

In essence, the 3D convolution is the extension of 2D convolution. The 3D-CNN+RNN architecture proposed in this work consists of 3 convolutional layers. We use a receptive field of $3 \times 3 \times 3$ followed by a max pooling operation for every convolutional layer. The activation function is Rectified linear unit (ReLU). A batch normalization layer [22] was employed for all the convolutional layers. Dropout with rate of 0.5 was employed in convolutional layers. The weights are initialized with Xavier initialization [23]. We employed multiple gated recurrent units (GRU) modules [24] where each feature map of the last convolutional layer is fed to the GRU [25]. Hence, we had a total of 16 separate GRU modules for 16 filters at the last convolutional layer output. We constructed 25 recurrent layers for each feature map, where 25 is the number of time steps mapped from the 200 time steps in the original spectrogram. We used recurrent networks with 32 GRU cells. The output for each RNN (many-to-one configuration) is concatenated and then fed into a fully connected layer. The combined 3D-CNN and RNN are optimized jointly by employing backpropagation algorithm. A softmax layer with two nodes is used (bird vs non-bird). The network is trained using RMSProp optimizer [26] with momentum of 0.9 and initial learning rate of 10^{-3}. We used batches of 8 training example to train our models. The categorical cross-entropy is used as a loss function. Tensorflow [27] is used to implement the models. The code to reproduce the results is made available in `https://github.com/himaivan/BAD2`.

4. EXPERIMENTS

4.1. Evaluation metric

The performance evaluation metric for bird sound classification is reported in terms of Area Under the ROC curve (AUC) as suggested in the challenge plan.

4.2. Baseline 2D CNN+RNN

The state-of-the-art deep learning method (CNN+RNN) has been employed in many audio classification tasks. We trained a CNN+RNN to be used as a baseline and to understand the benefit of 3D convolution. Instead of 3D features input, the input to CNN+RNN is a 2D log Mel-spectrogram image (80×400, over 10-second). We used a window of 50 ms for STFT analysis and 50% overlap. The CNN+RNN architecture consists of 3 convolutional layers and ReLU is used as an activation function. We use a receptive field of 3×3 with max-pooling sizes after each convolutional layer 2×2, stride of 2. We employed recurrent networks with 64 GRU cells. The RNN output is followed by a fully connected layer.

4.3. Training

We tested different parameter combinations to decide the final architecture to be used in the evaluation which include the number of CNN layers $\{3, 4\}$, drop-out rates $\{0.5, 0.7\}$, and the number of GRU cells $\{16, 32\}$. We also tested mean-pooling over time and max-pooling over time on the RNN outputs [28], and the use of convolutional attention module to learn bird calls structures relevant to the task [29]. However, we did not find that the pooling strategy and to include such attention mechanism improve the overall performance, and further investigation is necessary. For the first training strategy, we trained our baseline model using 97% of the total data. The 3% validation split is used to monitor the training process and for selecting final models. We stopped the training after 150 epochs to avoid overfitting. Figure 2 shows that a plateau is reached after about 60 epochs, and then continue to decrease. The training time is approximately 41 hours in our implementation using a Tesla M40 GPU. Since the training data is large, we did not perform data augmentation strategy. We then selected 5 models from different epoch with the highest accuracy on the validation split and averaged the predictions. We also trained our model using 3-way cross-validation strategy where in each fold two sets were used for training and the other one for testing, and averaged the predictions (hence, 15 networks were selected, five models for each cross-validation fold).

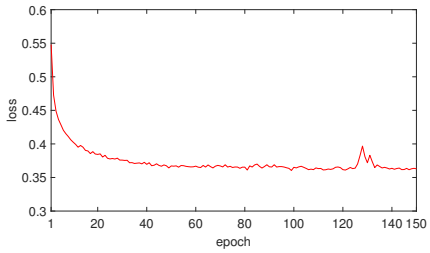


Figure 2: Training losses as a function of the training epochs.

5. RESULTS

Table 2: Stratified 3-way cross-validation results.

Train Configuration	Test	AUC
freefield1010 + warblrb10k	BirdVox	63.1%
freefield1010 + BirdVox	wabrlrb10k	85.9%
warblrb10k + BirdVox	freefield1010	79.4%
model ensemble	Evaluation data	88.7%

Our proposed 3D-CNN+RNN obtained a preview score of 87.13% when model is trained using the combined data (by averaging the predictions of five models from different epoch). Note that evaluating one model achieved 86.72%. Selecting one robust model is still applicable, for example, when such model is deployed in a hardware with limited processing power. In contrast, it is often not practical to perform a model ensemble method even though it improves the classifier performance in most cases. Meanwhile, our 2D CNN+RNN baseline obtained 83.15% AUC score. The 3-way cross-validation results where in each fold two sets were used for training and the other one for testing obtain 88.70% AUC score on the unseen evaluation data (via model ensemble method). For a comparison, training three instances of networks using the combined data with different weight initialization and averaging the predictions obtained 88.64%.

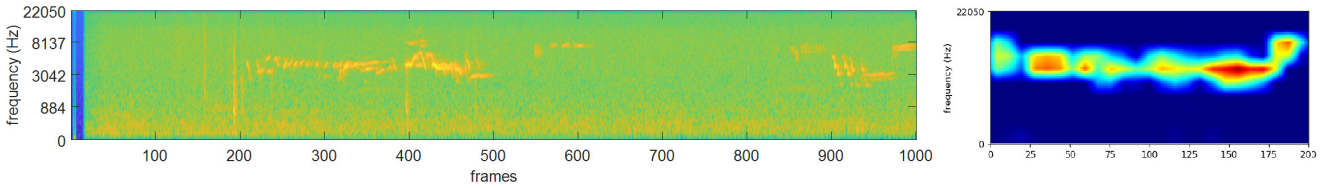


Figure 3: Original spectrogram (concatenation of 5, 80×200) for positive class (left) and the Grad-CAM visualization (right), using a 3D-CNN+RNN model. The red regions correspond to high score for class.

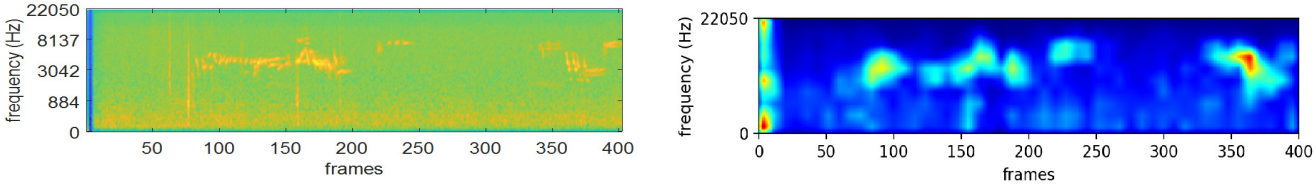


Figure 4: Original spectrogram (80×400) for positive class (left) and the Grad-CAM visualization (right), using a CNN+RNN model. The red regions correspond to high score for class.

We also tested pseudo-labeling approach inspired by the work of *Saito et al.*, 2017 [30]. To improve the accuracy of predicting pseudo-labels from unlabeled target samples, we used multiple networks simultaneously to work as predictor. A new target sample is selected if it satisfies two conditions: (1) All predictors predict the same label, and (2) All predictors achieve a confidence score exceeding the threshold. We adapted only the last layer of the trained network (while freezing the weights of other layers) using only evaluation data which have been annotated with pseudo-labels. However, we did not find any improvement with this model adaptation method.

5.1. Model Adaptation

For this challenge, the organizers have revealed that the evaluation dataset consists of 2,000 recordings from the same conditions as the *warblrb10k* data. To improve the model, we performed model adaptation where we adapted a model trained with freefield1010 and BirdVox data with *wabrlrb10k* data (by re-training only the last layer of the network). This results in 89.4% AUC score from 85.9% in Table 2. This model is then used to evaluate only the test set with the same condition as *warblrb10k*. We then used this new score instead of the prediction from our best (ensemble) model (88.70%) for the 2,000 recordings of *warblrb10k* data. This yielded our best result for this challenge with a preview score of **89.58%** AUC score.

5.2. Visualization

Recently, several techniques have been proposed to identify pattern and visualize the impact of the particular regions that are important for the model to make a prediction. This work adopted a Gradient-weighted Class Activation Mapping (Grad-CAM) [31, 32] to visualize our trained model. The Grad-CAM computed the gradient of the predicted score for a particular class with respect to feature maps output of a final convolutional layer. The result highlights the importance of feature maps for a target class. This method does not require architectural changes or re-training in order to generate visual explanations from any CNN-based networks. Note that the feature map activation at the last 3D convolutional layer is a 2D image which is mapped from a 3D input. Hence, it may not be straightforward to determine frame-based correspondence in the temporal axis between the Grad-CAM image and the spectrogram input. Nevertheless, as shown in Figure 3, the 3D convolution highlights only frequency bands where the bird calls are located across the temporal dimension. As a comparison, the 2D convolution in CNN+RNN highlights few specific locations of the bird calls, and include low-frequency regions with no bird calls. This shows that 3D convolution is more capable of extracting in terms of long-term time information in bird calls.

6. CONCLUSION

This paper proposed 3D convolutional recurrent neural networks for bird audio detection challenge. Our results show that a redundancy in the long-term time modeling of bird sounds can be exploited using both 3D convolution and recurrent layers. The proposed architecture is preferred compared to a conventional CNN+RNN technique. Building a robust deep learning model typically requires a large amount of labeled training data. However, obtaining large amounts of data is an expensive task and not always feasible. In future work, we will investigate the method of generating labeled data via a pseudo-labeling method where approximate labels are produced from unlabeled data. This can be achieved, for example, using generative adversarial networks. Domain adaptation using adversarial learning is another alternative to build a discriminative model and invariant to domain at the same time.

7. REFERENCES

[1] G. R. Walther et al., "Ecological responses to recent climate change," *Nature*, vol. 416, no. 6879, pp. 389–395, 2002.

[2] M. Towsey et al., "The use of acoustic indices to determine avian species richness in audio-recordings of the environments," *Ecological Informatics*, vol. 21, pp. 110–119, 2014.

[3] J. Sueur and A. Farina, "Ecoacoustics: the ecological investigation and interpretation of environmental sound," *Biosemiotics*, vol. 8, no. 3, pp. 493–502, 2015.

[4] D. Stowell et al., "Bird detection in audio: a survey and a challenge," in *Proceedings of IEEE International Workshop on Machine Learning for Signal Processing*, 2016, pp. 1–6.

[5] C. Szegedy et al., "Going deeper with convolutions," in *Proceedings of Computer Vision and Pattern Recognition*, 2014, pp. 1–9.

[6] K. He et al., "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, no. 770-778, 2016.

[7] G. Huang et al., "Densely connected convolutional networks," in *Proceedings of Computer Vision and Pattern Recognition*, 2017, pp. 4700–4708.

[8] D. Stowell and M. D. Plumbley, "Automatic large-scale classification of bird sounds is strongly improved by unsupervised feature learning," *PeerJ:e488*, 2014.

[9] P. R. Ehrlich et al., "Birds of Standford essays," Available online: https://web.stanford.edu/group/stanfordbirds/text/uessays/uhelp_essay_list.html.

[10] S. Ji et al., "3D convolutional neural networks for human action recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2013.

[11] A. Torfi et al., "3D convolutional neural networks for cross audio-visual matching recognition," *IEEE Access*, vol. 5, pp. 22 081–22 091, 2017.

[12] A. Torfi, J. Dawson, and N. M. Nasrabadi, "Text-independent speaker verification using 3D convolutional neural networks," in *Proceedings of IEEE International Conference on Multimedia and Expo*, 2018.

[13] I. Teivas, "Video event classification using 3D convolutional neural networks," Master's thesis, Tampere University of Technology, 2016.

[14] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Advances in neural information processing systems*, pp. 1097–1105, 2012.

[15] M. Lasseck, "Towards automatic large-scale identification of birds in audio recordings," in *Proceedings of Experimental IR Meets Multilinguality, Multimodality, and Interaction - 6th International Conference of the CLEF Association, CLEF*, 2015, pp. 364–375.

[16] I. Potamitis, "Unsupervised dictionary extraction of bird vocalisations and new tools on assessing and visualising bird activity," *Ecological Informatics*, vol. 26, pp. 6–17, 2015.

[17] T. Pellegrini, "Densely connected cnns for bird audio detection," in *Proceedings of the European Signal Processing Conference (EUSIPCO)*, 2017, pp. 1784–1788.

[18] E. Cakir et al., "Convolutional recurrent neural networks for bird audio detection," in *Proceedings of the European Signal Processing Conference (EUSIPCO)*, 2107, pp. 1794–1798.

[19] S. Adavanne et al., "Stacked convolutional and recurrent neural networks for bird audio detection," in *Proceedings of the European Signal Processing Conference (EUSIPCO)*, 2017, pp. 1779–1783.

[20] D. Stowell et al., "Automatic acoustic detection of birds through deep learning: the first bird audio detection challenge," *Methods in Ecology and Evolution*, 2018.

[21] M. Towsey et al., "A toolbox for animal call recognition," *Bioacoustics : The International Journal of Animal Sound and its Recording*, vol. 21, no. 2, pp. 107–125, 2012.

[22] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[23] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pp. 249–256, 2010.

[24] K. Cho et al., "Learning phrase representations using RNN encoder-decoder for statistical machine translation," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014, p. 17241734.

[25] H. Harutyunyan and H. Khachatrian, "Combining CNN and RNN for spoken language identification," Retrieved from https://yerevann.github.io/2016/06/26/combining-cnn-and-rnn-for-spoken-language-identification/, 2016.

[26] T. Tieleman and G. Hinton, "Lecture 6.5—RMSProp: Divide the gradient by a running average of its recent magnitude," COURSERA: Neural Networks for Machine Learning, 2012.

[27] M. Abadi et al., "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," *arXiv:1603.04467*, 2016.

[28] S. Mirsamadi et al., "Automatic speech emotion recognition using recurrent neural networks with local attention," in *Proceedings of IEEE International Conference on Acoustic, Speech, and Signal Processing*, 2017, pp. 2227–2231.

[29] C.-W. Huang and S. Narayanan, "Deep convolutional recurrent neural network with attention mechanism for robust speech emotion recognition," in *Proceedings of the IEEE International Conference on Multimedia and Expo*, 2017, pp. 583–588.

[30] K. Saito, Y. Ushiku, and T. Harada, "Asymmetric tri-training for unsupervised domain adaptation," in *arXiv preprint arXiv:1702.08400*, 2017.

[31] R. R. Selvaraju et al., "Grad-CAM: Visual explanations from deep networks via gradient-based localization," in *Proceedings of IEEE International Conference on Computer Vision*, 2017, pp. 618–626.

[32] I. Kim, "Grad-CAM-tensorflow," Available online: https://github.com/insikk/Grad-CAM-tensorflow/, 2018.

AUDIO FEATURE SPACE ANALYSIS FOR ACOUSTIC SCENE CLASSIFICATION

Tomasz Maka

West Pomeranian University of Technology, Szczecin
Faculty of Computer Science and Information Technology
Zolnierska 49, 71-210 Szczecin, Poland
tmaka@wi.zut.edu.pl

ABSTRACT

The paper presents a study of audio features analysis for acoustic scene classification. Various feature sets and many classifiers were employed to build a system for scene classification by determining compact feature space and using an ensemble learning. The input feature space containing different sets and representations were reduced to 223 attributes using the importance of individual features computed by gradient boosting trees algorithm. The resulting set of features was split into distinct groups partly reflected auditory cues, and then their contribution to discriminative power was analysed. Also, to determine the influence of the pattern recognition system on the final efficacy, accuracy tests were performed using several classifiers. Finally, conducted experiments show that proposed solution with a dedicated feature set outperformed baseline system by 6%.

Index Terms— audio features, auditory scene analysis, ensemble learning, majority voting

1. INTRODUCTION

The classification of the acoustical environment plays an essential role in human-machine interaction systems and it becomes a very popular research area in the last decade. The process of acoustical scene analysis involves many auditory cues [1] to determine the components of the scene. These cues are exploited to decompose and grouping of acoustic streams based on perceptual mechanisms of human hearing [2]. Attributes like periodicity, onsets and offsets, amplitude and frequency modulation, discontinues in the frequency domain, time–frequency units are very often used in the process of forming auditory objects. The time–frequency structure of an acoustic scene is dependent on the number of sound sources, its properties and variability in time. Additionally, the knowledge of acoustical attributes and their perceptual and physical meaning allows to create more sophisticated features and facilitate the scene decomposition.

On the other hand, in the deep–learning paradigm [3] the features are computed using unsupervised learning with only minimal preprocessing of the raw audio data as an input. In particular, features estimated in convolutional neural networks [4] yield to high classification accuracy and considerably outperforms the traditional pattern recognition systems. The problem with such features is the difficulty in their acoustical interpretation which may be necessary for system adaptation to changing environmental conditions in the acoustical scene. The solution in such a case requires a lot of data, causing the model to become large and complex. Since the features are the critical element of audio analysis systems, its selection is not a trivial task. The type of features and the size of the feature space determine the model used at the pattern recognition stage. Model complexity directly affects the system implementation, it defines required memory, computational resources and is the component influencing on the classification accuracy.

The most audio analysis systems dedicated to events detection or scene classification and using low–level features generate large feature spaces often containing more than a thousand attributes. Authors in [5] proposed a system with a large number of cepstral, spectral, voicing and energy features with statistical functionals, delta and acceleration coefficients. The parametrisation stage operated on the feature space with 6669 elements. The approach to event detection described in [6] uses 4096 audio features derived from well–known MFCC [7] features. An approach using 2000 features based on non-negative supervised matrix factorisation with Gaussian kernel SVM classifier is presented in [8]. A dimensionality of feature space equal to 4096 with were proposed in [9]. The computed random features approximating three types of kernels with SVM classifier were used to acoustic scene classification task. A very low-dimensional feature space was presented in [10]. Only nine optimised AMS (Amplitude Modulation Spectrum) features together with the LDA classifier was employed to classify acoustic scenes.

This study is a part of the work being developed for the purpose of creating the hierarchy of the robust audio features and its high–level representations for extracting objects and their properties from an audio stream.

2. PROPOSED FRAMEWORK

Due to the attempt of audio features analysis dedicated to acoustic scene analysis, we decided to use the traditional machine learning scheme. Such an approach is realised in two steps, where at first stage an input signal is converted to a feature space, then the data is fed to the classifier at the second step. The initial set of features was inspired by the auditory cues proposed for scene analysis [1, 2]. Dimensionality of the source set of features was equal to 2861. Next, the number of attributes was reduced in the feature importance analysis process using Gradient Boosting Machine [11] for whole development dataset. The resulting feature vector is composed of 13 subsets with 223 discriminative attributes as depicted in Figure 1. The final subsets can be briefly summarised as follows:

Binaural unit (F_1) – interaural time difference, interaural intensity difference, interaural coherence, and azimuth.

Pitch properties (F_2) – statistical properties of pitch contour.

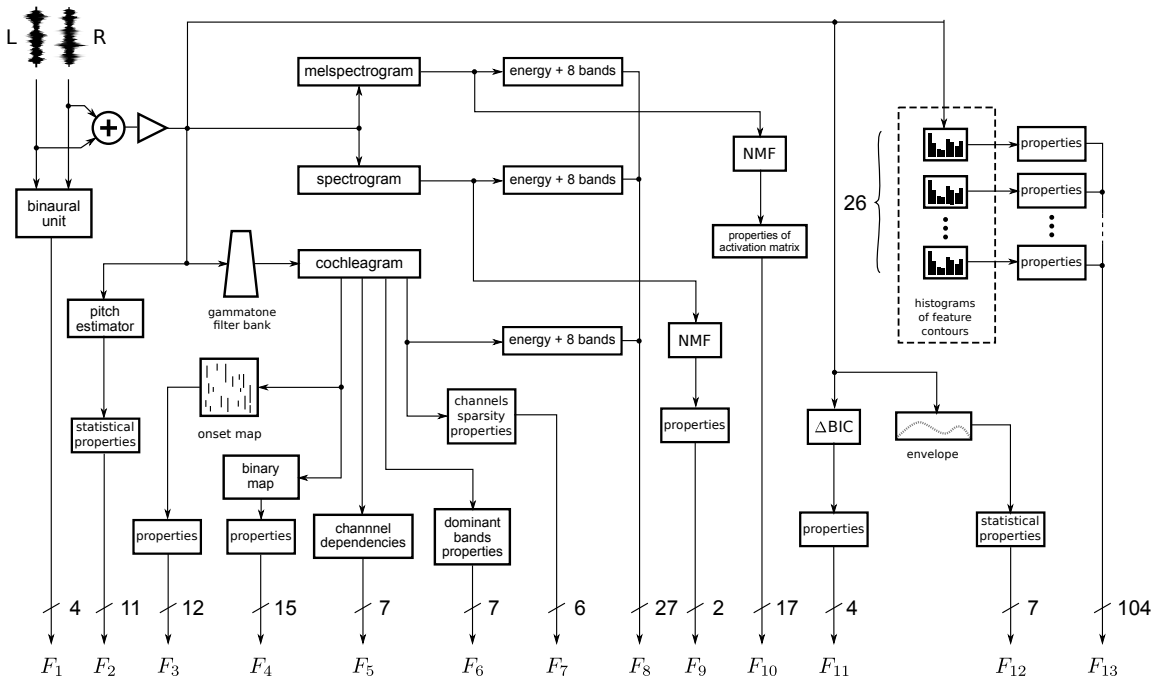


Figure 1: The diagram of the proposed system converting an audio signal to feature vector with 223 attributes.

Onset map (F_3) – properties of onsets detected in all channels of cochleagram.

Binary map (F_4) – attributes of binary map obtained by thresholding single channels of cochleagram.

Channel dependencies (F_5) – energy differences between neighbouring channels of cochleagram.

Dominant bands (F_6) – selected number of bands with the highest energies in cochleagram [12].

Channels sparsity (F_7) – Hoyer sparsity [13] computed for the individual channels of cochleagram.

Sub-band energies (F_8) – energies calculated in 8 equally sized ranges of cochleagram, melspectrogram and spectrogram.

Spectrogram activations (F_9) – attributes of activation matrix by computing non-negative matrix factorisation of spectrogram.

Melspectrogram activations (F_{10}) – properties of activation matrix by computing non-negative matrix factorisation of melspectrogram.

ΔBIC trajectory (F_{11}) – attributes of trajectory calculated as a difference between Bayesian Information Criterion (BIC) values of models used in audio segmentation [14].

Temporal envelope (F_{12}) – properties of temporal envelope [15].

Histograms of feature contours (F_{13}) – characteristic of histograms obtained for various [16] low–level feature contours.

In order to determine the variability of attributes between classes, we have averaged feature vectors over development set and mapped them clockwise onto unit circle as shown in Figure 2. Such visualisation highlights the similarities and differences between classes and can be used to determine the discriminative attributes. For example, in the case of the 'Tram' and 'Bus' classes the averaged feature vectors are quite similar and may be a reason for misclassification. After initial experiments with obtained feature space and

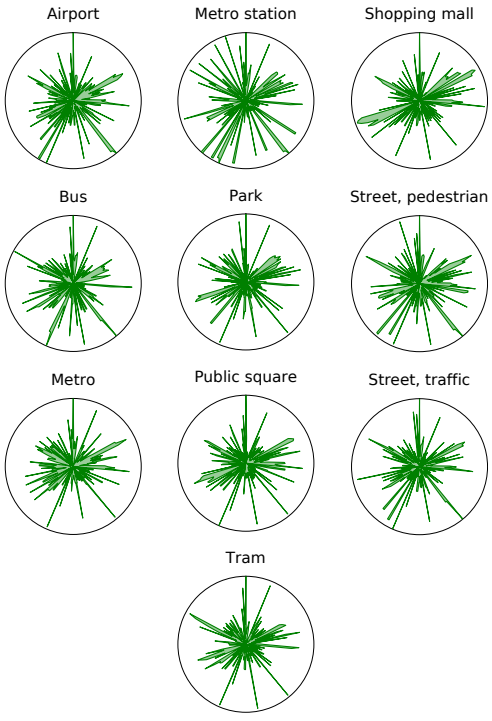


Figure 2: Averaged, normalised and mapped onto unit circle feature vectors for the whole development set.

standard classifiers, we decided to employ an ensemble learning. The reason was the classification accuracies we acquired for 64 individual classifiers because the average accuracy value was close to the baseline. The selection of classifiers was performed from a set of the classifiers with the accuracies higher than 50%. Then, an ensemble learning with majority/hard voting was executed. In the next step, successive classifier combinations were removed or replaced from the set to maximise the accuracy. The procedure ends when no improvements in classification accuracy occur.

In the result, a set of classifiers presented in Table 1 have been used in the majority voting scheme. There was no specific tuning of the classifiers, the parameters and configurations were selected randomly by the selection algorithm.

Table 1: The final set of classifiers in the majority voting scheme.

Classifier	Description
C_1	Linear Discriminant Analysis
C_2	Quadratic Discriminant Analysis
C_3	Random Forest classifier with 10 trees using Gini impurity as splitting metric.
C_4	Random Forest classifier with 100 trees using Gini impurity as splitting metric.
C_5	Random Forest classifier with 100 trees using entropy to compute information gain.
C_6	Multi-layer perceptron classifier. It uses 3 hidden layers with 30 hidden units each.
C_7	K-nearest neighbors classifier with K=20.
C_8	Bagging classifier with 500 linear support vector classification estimators.

3. EXPERIMENTAL EVALUATION

The system performance was evaluated on the development dataset of DCASE'2018 competition (Task 1) [17]. The audio data was recorded in 10 acoustic scenes and consists of binaural, 8640 segments each 10 seconds long using 48 kHz sampling rate and 24-bit resolution. The recordings were captured in six European cities.

In the first experiment, we have verified classification accuracy for individual classifiers using a complete feature set. The results are presented in Figure 3. In three cases, the accuracy exceeded 60%: for classifier C_1 is equal 62.9%, for C_4 is 63.2% and for classifier C_5 is equal to 61.9%.

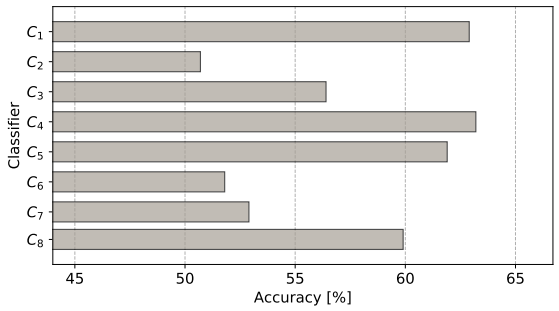


Figure 3: Influence of individual classifiers on the classification accuracy using the complete feature space.

Another experiment was to determine a discriminate power for subsets ($F_1 - F_{13}$) defined in Figure 1. The performance of acoustic scene recognition for individual subsets is presented in Figure 4.

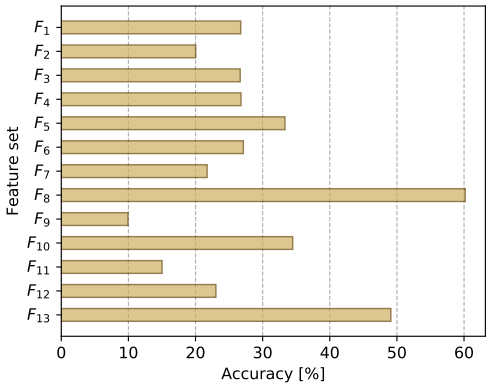


Figure 4: System performance using separate feature subsets.

The results show that the most discriminative subset F_8 gives the accuracy equal to 60.17%. Because this subset includes attributes from three different time-frequency representations in various frequency ranges, we examined the influence of each of the representations on the classification effectiveness. In Table 3 the results for three feature vectors computed from different representations are depicted. For each case the frequency range in further divided into eight equal bands to form the feature vector. The best-obtained accuracy is observed for cochleagram which may suggest that most discriminative data is located below 8kHz in the frequency domain.

The impact of individual subsets on the classification effectiveness was carried out in two subsequent experiments. In the first analysis, we started with the subset that has the highest discriminatory power (see Figure 4), then the main set was increased by the consecutive subsets as shown in Figure 5.

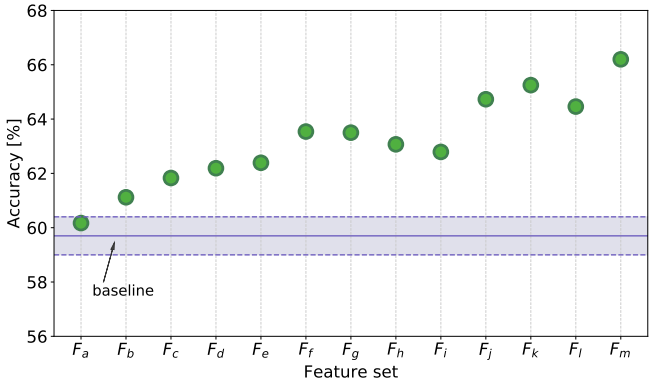


Figure 5: Classification results with combined subsets of attributes in order from the most to the least discriminative[1].

[1] $F_a = F_8$; $F_b = F_a \cup F_{13}$; $F_c = F_b \cup F_{10}$; $F_d = F_c \cup F_5$; $F_e = F_d \cup F_6$; $F_f = F_e \cup F_4$; $F_g = F_f \cup F_1$; $F_h = F_g \cup F_3$; $F_i = F_h \cup F_{12}$; $F_j = F_i \cup F_7$; $F_k = F_j \cup F_2$; $F_l = F_k \cup F_{11}$; $F_m = F_l \cup F_9$

Table 2: The class-wise accuracy of the development set: confusion matrix (a), comparison with the baseline (b).

Category \ Estimate	Airport	Bus	Metro	Metro station	Park	Public square	Shopping mall	Street, pedestrian	Street, traffic	Tram
Airport	**47.2**		2.6	4.9	0.4	8.3	18.1	18.5		
Bus		**63.6**	5.4		1.2	0.4				29.3
Metro		5.4	**60.5**	9.2		1.1		0.4	3.4	19.9
Metro station	6.2	1.2	7.7	**55.6**	1.2	2.3	3.9	10.7	4.6	6.6
Park			0.4	2.1	**88.8**	3.7		3.3	0.9	0.8
Public square	0.5	1.9	1.3	3.7	16.7	**38.9**	1.9	19.0	14.7	1.4
Shopping mall	5.0			1.1		0.7	**89.6**	3.6		
Street, pedestrian	4.0			0.9	3.2	20.6	6.5	**57.5**	5.7	1.6
Street, traffic			0.4	1.6		5.3		6.5	**86.2**	
Tram	1.1	9.2	11.9	1.1			1.9	0.8		**73.9**

(a)

Scene class	Accuracy	
	Baseline	Proposed
Airport	72.9 %	47.2 %
Bus	62.9 %	63.6 %
Metro	51.2 %	60.5 %
Metro station	55.4 %	55.6 %
Park	79.1 %	88.8 %
Public square	40.4 %	38.9 %
Shopping mall	49.6 %	89.6 %
Street, pedestrian	50.0 %	57.5 %
Street, traffic	80.5 %	86.2 %
Tram	55.1 %	73.9 %
Average	**59.7 %** (+/- 0.7)	**66.2 %**

(b)

Table 3: Performance of individual representations of set F_8.

Representation	Frequency range [Hz]	Bands	Accuracy
Cochleagram	50 – 8000	128	51.99 %
Melspectrogram	0 – 12000	128	46.58 %
Spectrogram	0 – 24000	1024	42.65 %

In the second experiment, the attempts were made to remove further subsets to assess the impact of the resulting feature set on the classification effectiveness. According to the results depicted in Figure 6, the subset F_8 is a crucial part of the feature vector. Its contribution is similar as in case of the experiment which results are presented in Figure 4. Interestingly, the smallest decreasing of accuracy is observed for the pitch related features (subset F_2) although such attributes are discriminative for parts contained speech in an audio signal.

Finally the classification experiments were performed using the proposed framework and development dataset. The confusion matrix is presented in Table 2a. The best result was obtained for 'Shopping mall' (89.9%) and the worst for 'Public square' (38.9%) with overall system performance equal to 66.2%. According to the confusion matrix, analogies can be noticed between classes with sound sources sharing similar physical properties. For example, such a situation is visible for classes 'Bus', 'Metro' and 'Tram'. The comparison of our system with the baseline is shown in Table 2b, where in case of classes 'Airport' and 'Public square' decrease in accuracy was observed.

Due to the length of the recordings, many of the segments have a similar acoustical structure for different classes which caused misclassification. Unfortunately, is such case, low-level audio features are ineffective.

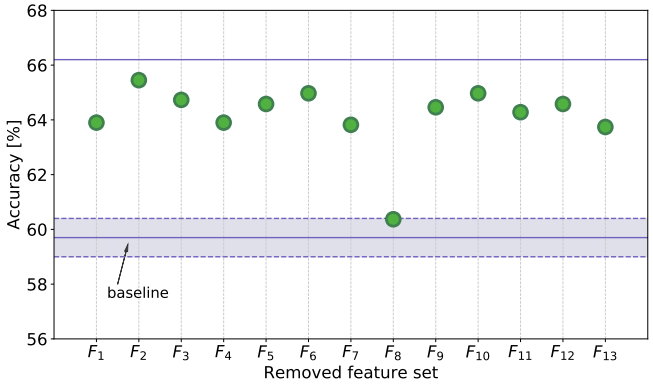


Figure 6: Obtained accuracy in the situation of removing individual subsets of the final feature set.

4. CONCLUSION

In this paper, we have presented an approach to classify of acoustic scenes with the dedicated set of audio features and ensemble learning classification stage. An advantage of our system is a small set of features which can be used in systems with low resources. At the current stage of development, our system has worse efficiency in comparison to deep-learning based solutions, but similar to human hearing abilities for the development set. In future work, we intend to design hierarchical audio features dedicated to specific acoustic scenes including events and background noise. For this purpose, we have designed and implemented a dedicated application[2]. It can be used to browse various audio representations and to support the process of developing a new hybrid features.

[2] http://quefrency.org/dcase2018

5. REFERENCES

[1] D. Wang and G. J. Brown, *Computational Auditory Scene Analysis – Principles, Algorithms, and Applications*, D. Wang and G. J. Brown, Eds. IEEE Press / Wiley–Interscience, 2006.

[2] A. S. Bregman, *Auditory Scene Analysis – The Perceptual Organization of Sound*, 1st ed. The MIT Press, 1994.

[3] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, May 2015.

[4] Y. LeCun, K. Kavukcuoglu, and C. Farabet, "Convolutional networks and applications in vision," in *IEEE International Symposium on Circuits and Systems – ISCAS'2010*. IEEE, 2010, pp. 253–256.

[5] J. T. Geiger, B. Schuller, and G. Rigoll, "Large-scale audio feature extraction and SVM for acoustic scene classification," in *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics – WASPAA'2013*, New Paltz, NY, USA, October 20–23 2013, pp. 1–4.

[6] F. Metze, S. Rawat, and Y. Wang, "Improved audio features for large-scale multimedia event detection," in *IEEE International Conference on Multimedia and Expo – ICME'2014*. IEEE, 2014, pp. 1–6.

[7] S. B. Davis and P. Mermelstein, "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences," *IEEE Transactions on Acoustics Speech and Signal Processing*, vol. ASSP-28, no. 4, pp. 357–366, August 1980.

[8] A. Rakotomamonjy, "Enriched supervised feature learning for acoustic scene classification," Normandie Universite, Tech. Rep., 2016.

[9] A. Jimenez, B. Elizalde, and B. Raj, "Dcase 2017 task 1: Acoustic scene classification using shift-invariant kernels and random features," *arXiv preprint arXiv:1801.02690*, 2018.

[10] S. Agcaer, A. Schlesinger, F.-M. Hoffmann, and R. Martin, "Optimization of amplitude modulation features for low-resource acoustic scene classification," in *23rd European Signal Processing Conference – EUSIPCO'2015*, Nice, France, August 31 – September 4 2015, pp. 2556–2560.

[11] J. H. Friedman, "Greedy function approximation: A gradient boosting machine," *The Annals of Statistics*, vol. 29, no. 5, pp. 1189–1232, 2001.

[12] T. Maka, "Auditory scene classification based on the spectro-temporal structure analysis," West Pomeranian University of Technology, Szczecin, Tech. Rep., 2017.

[13] P. O. Hoyer, "Non-negative matrix factorization with sparseness constraints," *Journal of Machine Learning Research*, vol. 5, no. Nov, pp. 1457–1469, November 2004.

[14] S. S. Chen and P. Gopalakrishnan, "Speaker, environment and channel change detection and clustering via the bayesian information criterion," in *DARPA Broadcast News Transcription and Understanding Workshop*, Lansdowne, Virginia, USA, February 8–11 1998, pp. 127–132.

[15] J. Gillard and M. Schutz, "The importance of amplitude envelope: Surveying the temporal structure of sounds in perceptual research," in *10th Sound and Music Computing Conference – SMC'2013*, Stockholm, Sweden, July 30 – August 3 2013, pp. 62–68.

[16] A. Lerch, *An Introduction to Audio Content Analysis – Applications in Signal Processing and Music Informatics*. John Wiley & Sons, Inc., 2012.

[17] A. Mesaros, T. Heittola, and T. Virtanen, "A multi-device dataset for urban acoustic scene classification," 2018, submitted to DCASE2018 Workshop.

DNN BASED MULTI-LEVEL FEATURE ENSEMBLE FOR ACOUSTIC SCENE CLASSIFICATION

Jee-weon Jung, Hee-soo Heo*, Hye-jin Shim, and Ha-jin Yu*†

School of Computer Science, University of Seoul, South Korea

ABSTRACT

Various characteristics can be used to define an acoustic scene, such as long-term context information and short-term events. This makes it difficult to select input features and pre-processing methods suitable for acoustic scene classification. In this paper, we propose an ensemble model that exploits various input features in which the strength for classifying an acoustic scene varies: i-vectors are used for segment-level representations of long-term context, spectrograms are used for frame-level short-term events, and raw waveforms are used to extract features that could be missed by existing methods. For each feature, we used deep neural network based models to extract a representation from an input segment. A separated scoring phase was then exploited to extract class-wise scores on a scale of 0 to 1 that could be used as confidence measures. Scores were extracted using Gaussian models and support vector machines. We tested the validity of the proposed framework using task 1 of detection, and classification of acoustic scenes and events 2018 dataset. The proposed framework had an accuracy of 73.82% for the pre-defined fold-1 validation setup and 74.8% for the evaluation setup which is 7th in team ranking.

***Index Terms*—** Acoustic scene classification, DNN, raw waveform, i-vector

1. INTRODUCTION

There is an increasing demand for acoustic scene classification (ASC), a task that can be applied in various machines and intelligent systems. Three noticeable characteristics can be observed by analyzing the past editions of detection and classification of acoustic scenes and events (DCASE) competitions: (a) deep neural networks (DNNs) are mainly used with various architectures, (b) various features such as spectrograms, Mel frequency cepstral coefficients (MFCCs), and constant Q cepstral coefficients (CQCCs) [1] are used , and (c) ensemble of two or more classifiers are used with majority voting or score-sums.

Despite this active research, choosing appropriate features for ASC tasks remains difficult. One of the main factors complicating this problem may be the fact that different features are appropriate for representing each scene in an ASC task. For example, segment-level features such as i-vectors may be useful for classifying scenes where the characteristics appear over a long period of time. Frame-level features such as spectrograms can be used to classify scenes where events occur in a particular frequency band at short intervals. To consider the different characteristics that can define an acoustic scene, we trained DNNs that input each feature and agglomerate the results. Additionally, raw waveforms without feature extraction techniques can be input into the DNN to extract features internally with respect to ASC tasks during the training phase. By directly using raw waveforms, the DNN is expected to find most appropriate features for the target task.

Another problem is that the two methods most frequently used in ensembles of the DNNs (majority voting and score-sum) do not include confidence measures. In majority voting, the output of classifiers are voted, meaning that the precision ($\frac{true\ positive}{true\ positive + false\ positive}$) of the individual class of each system is not considered. Score-sum of DNN output layer uses a softmax activation as confidence score. This neglects the precision of classification on each class but also considered not ideal because in the case of softmax outputs, scores can be poorly calibrated [2]. Therefore, we added a separate scoring phase to calculate calibrated scores from trained DNNs [3].

In this paper, we make the following contributions:

1. Exploit various features including raw waveform that can be more useful for classifying acoustic scenes.
2. Train Gaussian models and support vector machines that inputs the output of DNN's code layer and extract scores with confidence.

Specifically, three features are individually studied for ASC task. The first feature is i-vector [4], a segment-level low dimensional representation, known to be suitable for ASC tasks. The second feature is a spectrogram, which is widely used for ASC task with convolutional neural networks (CNNs) [5], [6]. The last feature is a raw waveform, which is directly input into a DNN. We hypothesize that segment-level i-vectors can detect scenes using long-term context information, frame-level spectrogram can detect scenes involving short-term events, and raw waveforms can be used to find useful features for classifying acoustic scenes using DNN training. Single Gaussian models and support vector machines (SVMs) [7] use the outputs of DNN's code layers as input and are used as back-end classifiers to obtain confidence score for each class given an embedding. Final score is derived through score fusion using confidence scores. The overall proposed framework is depicted in Figure 1.

The remainder of this paper is organized as follows. Section 2 describes the three DNNs with different features, the back-end classifiers used in this study, and the ensemble methodology. The experimental settings and system specifications are presented in Section 3 with experimental results. The paper is concluded in Section 4.

2. SYSTEM DESCRIPTION

In this section, we describe the three systems in the ensemble according to their input features, the back-end classifiers used for scor-

*These authors have equal contribution.

† Corresponding author.

This research was supported by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Science, ICT & Future Planning(2017R1A2B4011609)

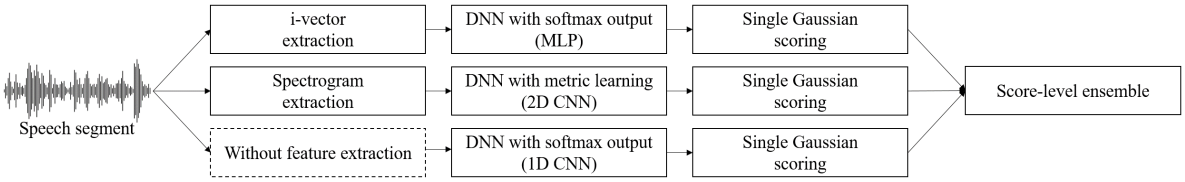


Figure 1: Illustration of the overall framework.

ing, and the ensemble methodologies.

2.1. i-vector based system

An i-vector (identity vector) is a low-dimensional representation of a given segment using factor analysis [4]. Regardless of the length of a given segment, one vector with fixed dimensionality is extracted. Originally, i-vectors were proposed for speaker verification, but in previous DCASE challenges, i-vectors have also performed well on ASC task [8, 9]. In this study, we used i-vectors as one of our input features, expecting that the segment-level representations would appropriately classify acoustic scenes defined by long-term contexts. The i-vector based DNNs is trained using a supervised training scheme with categorical cross-entropy objective functions and softmax activation.

2.2. Spectrogram based system

Spectrograms are widely used in audio signal processing systems, including speech recognition and speaker recognition. We hypothesized that this frame-level feature could be used to detect events occurring in a specific frequency band, and could therefore contribute to improving performance on ASC task. We used max feature map (MFM) based 2D CNN architecture to embed the spectrogram extracted from each segment [10]. In the MFM based architecture, instead of activation functions such as rectified linear units, a max operation is applied to multiple feature maps to calculate the output of each layer. In this study, we varied the filter sizes, expecting that the appropriate filter size would be found in the DNN training process through competition between filters.

The spectrogram-based system is trained using a metric learning scheme instead of conventional supervised training with softmax activation output layer. This learning scheme inputs two or more samples and trains the DNN to simultaneously decrease similarities between samples from different classes (= negative similarity) and increase similarities between samples from identical classes (= positive similarity). The cosine similarities are calculated between DNN embeddings at the code layer. Additionally, it has been shown that for performing the training of a DNN, it is more efficient for generalization to use similarities between an embedding and an average class embedding [11]. Therefore, the network is trained to minimize the loss defined by equation (1)

$$\mathcal{L} = \frac{1}{N_c}\sum_{i}^{N_c}\sum_{j\neq i}^{N_c}(CS(\boldsymbol{e}_i, \boldsymbol{m}_j) - CS(\boldsymbol{e}_i, \boldsymbol{m}_i)), \quad (1)$$

where, N_c is the number of classes, $\boldsymbol{e}_i$ is the embedding of a sample from the $i'th$ class, $\boldsymbol{m}_i$ is the average embedding of the $i'th$ class, and $CS(\cdot)$ is a cosine similarity operation between two embedding vectors. Figure 2 shows the process for calculating the positive and negative similarity samples defined in equation (1), based on ten classes. However, in datasets where the number of classes is small while the number of samples in each class is large, repeatedly calculating the average embeddings during the training process causes a large overhead.

Therefore, we calculated the average embeddings of each class at the beginning of the training as the centroid of each class and update it as the DNN training proceedes. The average embedding $\boldsymbol{m}_i^t$ of the $i'th$ class at time t is updated using equation (2)

$$\boldsymbol{m}_i^t = \alpha \boldsymbol{m}_i^{t-1} + (1-\alpha)\hat{\boldsymbol{m}}_i^t, \quad (2)$$

where,$\hat{\boldsymbol{m}}_i^t$ is the average embedding of class i calculated for each mini-batch, and α is momentum value which define a ratio between $\boldsymbol{m}_i$ and $\hat{\boldsymbol{m}}_i$.

Negative sampling is a technique that can effectively improve the performance of metric learning by searching hard negative cases [12]. In negative sampling, rather than using all samples, loss is calculated using samples that are relatively difficult to classify. However, negative sampling is time-consuming, and generally requires another classifier such as SVM solely for this operation. Instead of negative sampling, we used the modified loss shown in equation (3)

$$\mathcal{L}_{max} = \frac{1}{N_c}\sum_{i}^{N_c}\max_{\{0\leq j\leq N_c-1, j\neq i\}}(CS(\boldsymbol{e}_i, \boldsymbol{m}_j) - CS(\boldsymbol{e}_i.\boldsymbol{m}_i)), \quad (3)$$

In equation (3), positive similarities are used in the same way as the conventional loss defined in equation (1). On the other hand, only one of the $N_c - 1$ negative similarities is used to calculate loss, selected through max operations. Figure 3 shows an example of the operation of equation (3): the training process of $\mathbf{e}_1$ in which similarity with $\mathbf{m}_1$, the centroid of the same class, increases, and the similarity with $\mathbf{m}_2$, the centroid of the class that is most hard to classify, decreases. With such modifications, we expected that the DNN would be trained to better discriminate acoustic scenes with similar characteristics.

2.3. Raw waveform based system

Recently, promising results have been observed with DNNs that directly input raw waveforms. Such DNNs have been proposed for use with various tasks [13, 14, 15]. Through the visualization of raw-waveform-based DNN models, it has been shown that the kernels of 1D convolutional layers are trained to detect specific frequency bands [14]. Many raw waveform systems aim to extract

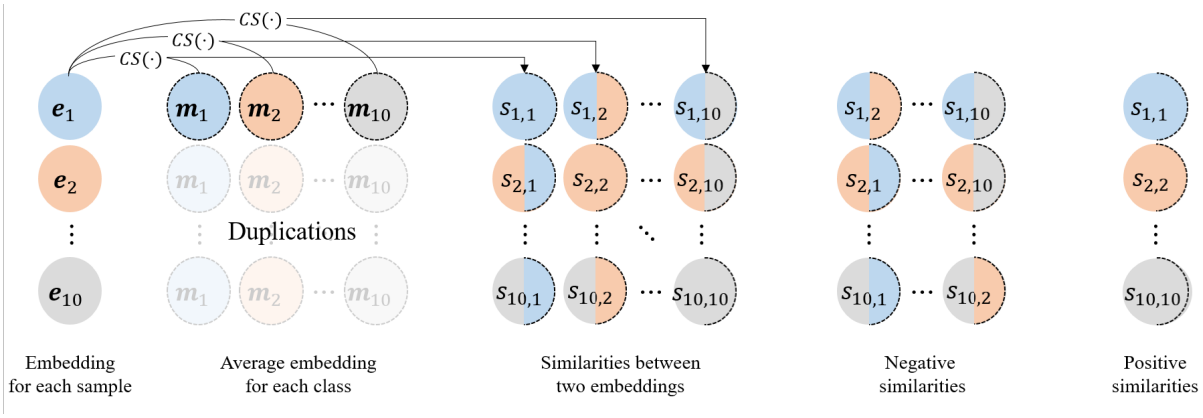


Figure 2: Concept illustration of the modified metric learning with learned mean embeddings of each class.

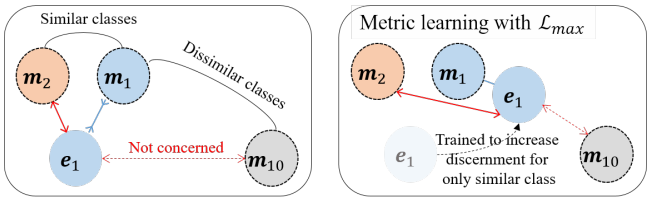


Figure 3: Concept illustration of equation (3) .

features that suit the objective defined by the loss function of DNN better than existing acoustic feature extracting techniques through extracting most useful frequency bands [14]. In this work, we use the RACNN-LSTM model proposed by Jung et al. [15] with a few modifications, considering the DCASE 2018 task 1 dataset. The raw waveform system that we used consists of convolutional blocks and fully connected layers: each convolutional block consists of a 1D convolutional layer, followed by batch normalization, rectified unit activation, and max pooling . The raw-waveform-based DNN is trained by supervised learning using a categorical cross-entropy loss function. Modifications and detailed descriptions of the raw-waveform-based system are present in Section 3.3.

2.4. Back-end scoring

Support vector machine (SVM) with RBF kernel and sigmoid kernel, single Gaussian model with diagonal and full covariance were used as back-end classifier. Classifiers were trained to discriminate acoustic scenes using DNN embeddings. In the spectrogram-based system, we used the code layer directly. The last hidden layer was used as the code layer for the i-vector and raw waveform systems. We expected that by using a back-end classifier for scoring instead of a softmax output, we could make the ensemble of multiple DNNs more efficient.

2.5. Ensemble method

Scores from each of the back-end scoring classifiers can be simply summed, because the scores already include the concept of confidence with a scale of zero to one. However, the different classifiers can have different discriminative powers for different acoustic scenes. To incorporate this concept, a precision vector is calculated based on the classification results for the validation dataset. The entries of a precision vector is the precision scores $\frac{true\ positive}{true\ positive + false\ positive}$ of each classifier for each class. Back-end classifier scores are multiplied by this precision vector before they are merged.

3. EXPERIMENTAL SETTINGS

Our experiments in this study used soundfile and scipy python modules for raw waveform and spectrogram extraction [16]. The Kaldi toolkit [17] was used for i-vector extraction. The Keras deep learning toolkit [18] with a tensorflow back-end [19, 20] was used for DNN training and decoding. The scikit-learn module was used for Gaussian model and SVM scoring [21].

3.1. Dataset

All experiments in this paper used task 1-a from the DCASE 2018 dataset [22]. Task 1-a in the DCASE 2018 dataset comprises 8,640 audio segments recorded in stereo at a 48 kHz sampling rate with 24 bit resolution and divided into 10 s lengths. Fourfold cross-validation was conducted using the provided meta data regarding recording locations. The development set and validation set do not use audio segments from identical locations. In this paper, we only report the accuracy of the first fold.

3.2. Feature configurations

We extracted i-vectors from a diagonal Gaussian mixture model (GMM) with 1024 components, trained with 60-dimensional MFCC features. A total variability matrix that can extract a 200-dimensional i-vector was trained for 10 iterations. Neither length normalization nor linear discriminant analysis were applied after i-vector extraction.

Spectrograms were extracted by shifting 30 ms windows by 10 ms. A spectrogram was represented by 721 coefficients for each window, and only 300 coefficients of low frequency bands were used; we empirically confirmed that low frequency bands are more

useful for ASC. Finally, a spectrogram of size 499×300 was extracted from each 10 s segment.

Stereo raw waveforms (with pre-emphasis) are used as input features to the DNN, resulting in feature shapes of $(48{,}000 \times 10, 2)$.

3.3. System configurations

The i-vector based DNN comprises 4 fully connected layers. In this system, the DNN acts only as a feature enhancer for the scoring step of the task, because the i-vector is already a sophisticated feature at the segment-level. The four fully connected layers each have 512 units, and L2 regularization is applied.

Spectrogram-based DNNs comprise two fully connected layers following three MFM layers. Fully connected layers contain 256 nodes activated by a leaky ReLU function [23]. L2 length normalization was applied to the output of the last fully connected layer following the work of Wan et al., who trained a DNN for speaker verification using metric learning [11]. The configuration of the MFM-based system is shown in Table 1. In each MFM layer, the output is calculated using the max operation between the feature maps generated by filters of different sizes. We simultaneously applied two types of pooling layers (max and average pooling) in the last CNN stage.

Table 1: Configuration of MFM based CNN system.

layer	output shape	kernel sizes
1^{st} MFM	499×300×32	5×5, 7×7, 9×9, 11×11
Max pooling	166×60×32	3×5
2^{nd} MFM	166×60×64	3×3, 5×5, 7×7, 9×9
Max pooling	55×12×64	3×5
3^{rd} MFM	55×12×64	3×3, 5×5, 7×7, 9×9
Max pooling	1×3×64	55×4
Average pooling	1×3×64	55×4
Concatenation	1×3×128	
Flatten	384	

Raw-waveform-based DNNs use the RACNN-LSTM model from Jung et al.'s work, with a few modifications [15]. Modifications include the following: the stride size of the strided convolutional layer was changed to 12 for a 48 kHz sampling rate, 256 kernels were used for the last convolutional layer, and stereo audio inputs were used instead of mono audio inputs.

3.4. Results

Experimental results for the provided fold 1 setup of the DCASE 2018 competition are presented in Table 2 in terms of classification accuracy. Each input feature is examined by four different classifiers, and the best results submitted to the DCASE 2018 competition are shown. The four columns of Table 2 each represent our submitted system for the DCASE 2018 competition, in which 'All' refers to the ensemble of single Gaussian and SVM classifiers.

Surprisingly, for the input features, raw waveforms performed the best, with an accuracy of 67.15 %. The three-feature ensemble increased accuracy more than 6 %. Although we do not show this result because of paper length limitations, the ensemble results for any two features improved performance in terms of classification accuracy. Therefore, we conclude that different features actually contribute to the ASC task based on the characteristics of each feature.

For back-end classifiers, we first compared the results of directly using softmax activation based classification to the results for separate scoring schemes using single Gaussian and SVM models. With the raw waveform as an input, conventional classification showed an accuracy of 64.71 %, while single Gaussian scoring and SVM scoring using the last hidden layer as code showed accuracies of 67.91 % and 66.56 %, respectively. Among the back-end classifiers, the accuracies of single Gaussian models were approximately 1 % higher, but noticeable differences were not measured. By applying a precision vector representing the accuracy of each acoustic scene system, we were able to improve performance when the precision vector was used with classifiers of the same type (e.g., diagonal Gaussian models and full Gaussian models). However, accuracy did not increase when the precision vector was used with different types of classifiers.

Table 2: Classification accuracy (%) for the individual systems and four-classifier ensemble system. The four columns indicate the four systems submitted to the DCASE task 1-a competition ('All w/o weight' is submission 1). 'w weight' refers to the case where classifier outputs were ensembled with the use of a precision vector. All refers to cases using two Gaussian and two SVM classifiers, Gaussian refers to cases using full and diagonal covariance classifiers, and SVM refers to cases using SVMs with RBF and sigmoid kernels.

system \ classifier	All w/o weight	All w weight	Gaussian w weight	SVM w weight
raw-waveform (val)	67.15	68.10	67.91	66.56
spectrogram (val)	66.24	66.20	66.44	66.44
i-vector (val)	63.74	63.93	65.17	63.66
Ensemble (val)	**73.82**	73.23	73.15	72.71
Ensemble (eval)	**74.8**	74.2	73.8	73.8

4. CONCLUSION AND FUTURE WORKS

Selecting appropriate features for each task is critically important for machine learning research. However, this is difficult because of the required domain expertise, such as knowledge regarding the characteristics of the input data and the understanding of the task to be performed. Segment-level i-vectors and frame-level spectrograms were used to detect both long-term contexts and short-term events, by training DNNs for each feature. Additionally, raw waveforms were used, with expectation that the kernel weights for 1D convolutional layers would be trained to extract the most discriminative features for each ASC task. We built an ensemble of DNNs with different input features, using score fusion on single Gaussian models and SVMs. An accuracy of 73.82 % and 74.8 % was shown for the DCASE 2018 task 1-a validation set and evaluation set, respectively.

In this study, we exploited multiple DNNs with different architectures, which respectively received different features. We then combined the results for each DNN. Training different types of features with a single DNN, however, may lead to the synergy of different features when training an integrated DNN. In the future, we plan to build a single integrated system that simultaneously receives multiple features. To achieve this type of system, we would need to simultaneously consider the various characteristics of different types of features.

5. REFERENCES

[1] M. Todisco, H. Delgado, and N. Evans, "Constant q cepstral coefficients: A spoofing countermeasure for automatic speaker verification," *Computer Speech & Language*, vol. 45, pp. 516–535, 2017.

[2] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, "On calibration of modern neural networks," *arXiv preprint arXiv:1706.04599*, 2017.

[3] S. Park, S. Mun, Y. Lee, and H. Ko, "Score fusion of classification systems for acoustic scene classification," *IEEE AASP Challenge on Detection and Classification of Acoustic Scenes and Events (DCASE)*, 2016.

[4] N. Dehak, P. J. Kenny, R. Dehak, P. Dumouchel, and P. Ouellet, "Front-end factor analysis for speaker verification," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 4, pp. 788–798, 2011.

[5] Z. Weiping, Y. Jiantao, X. Xiaotao, L. Xiangtao, and P. Shaohu, "Acoustic scene classification using deep convolutional neural network and multiple spectrograms fusion," in *Detection and Classification of Acoustic Scenes and Events 2017 Workshop (DCASE2017)*, 2017.

[6] E. Fonseca, R. Gong, D. Bogdanov, O. Slizovskaia, E. Gómez Gutiérrez, and X. Serra, "Acoustic scene classification by ensembling gradient boosting machine and convolutional neural networks," in *Virtanen T, Mesaros A, Heittola T, Diment A, Vincent E, Benetos E, Martinez B, editors. Detection and Classification of Acoustic Scenes and Events 2017 Workshop (DCASE2017); 2017 Nov 16; Munich, Germany. Tampere (Finland): Tampere University of Technology; 2017. p. 37-41.* Tampere University of Technology, 2017.

[7] B. Schölkopf, A. J. Smola, *et al.*, *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2002.

[8] B. Elizalde, H. Lei, G. Friedland, and N. Peters, "An i-vector based approach for audio scene detection," *IEEE AASP Challenge on Detection and Classification of Acoustic Scenes and Events*, 2013.

[9] J. Jung, H. Heo, I. Yang, S. Yoon, H. Shim, and H. Yu, "Dnn-based audio scene classification for dcase 2017: Dual input features, balancing cost, and stochastic data duplication," in *Detection and Classification of Acoustic Scenes and Events*, 2017.

[10] X. Wu, R. He, Z. Sun, and T. Tan, "A light cnn for deep face representation with noisy labels," *arXiv preprint arXiv:1511.02683*, 2015.

[11] L. Wan, Q. Wang, A. Papir, and I. L. Moreno, "Generalized end-to-end loss for speaker verification," *arXiv preprint arXiv:1710.10467*, 2017.

[12] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 815–823.

[13] D. Palaz, M. Doss, and R. Collobert, "Convolutional neural networks-based continuous speech recognition using raw speech signal," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2015, pp. 4295–4299.

[14] J. Lee, J. Park, K. Kim, Luke, and J. Nam, "Sample-level deep convolutional neural networks for music auto-tagging using raw waveforms," *arXiv preprint arXiv:1703.01789*, 2017.

[15] J. Jung, H. Heo, I. Yang, H. Shim, and H. Yu, "A complete end-to-end speaker verification system using deep neural networks: From raw signals to verification result," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018.

[16] E. Jones, T. Oliphant, P. Peterson, *et al.*, "SciPy: Open source scientific tools for Python," 2001–, [Online; accessed today]. [Online]. Available: http://www.scipy.org/

[17] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz, *et al.*, "The kaldi speech recognition toolkit," in *IEEE 2011 workshop on automatic speech recognition and understanding*, no. EPFL-CONF-192584. IEEE Signal Processing Society, 2011.

[18] F. Chollet *et al.*, "Keras," https://github.com/keras-team/keras, 2015.

[19] A. Martn, A. Ashish, B. Paul, B. Eugene, *et al.*, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," 2015. [Online]. Available: http://download.tensorflow.org/paper/whitepaper2015.pdf

[20] A. Martin, B. Paul, C. Jianmin, C. Zhifeng, D. Andy, D. Jeffrey, D. Matthieu, G. Sanjay, I. Geoffrey, I. Michael, K. Manjunath, L. Josh, M. Rajat, M. Sherry, M. G. Derek, S. Benoit, T. Paul, V. Vijay, W. Pete, W. Martin, Y. Yuan, and Z. Xiaoqiang, "Tensorflow: A system for large-scale machine learning," in *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, 2016, pp. 265–283. [Online]. Available: https://www.usenix.org/system/files/conference/osdi16/osdi16-abadi.pdf

[21] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[22] A. Mesaros, T. Heittola, and T. Virtanen, "A multi-device dataset for urban acoustic scene classification," 2018, submitted to DCASE2018 Workshop. [Online]. Available: https://arxiv.org/abs/1807.09840

[23] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *Proc. icml*, vol. 30, no. 1, 2013, p. 3.

DATA-EFFICIENT WEAKLY SUPERVISED LEARNING FOR LOW-RESOURCE AUDIO EVENT DETECTION USING DEEP LEARNING

Veronica Morfi

Machine Listening Lab,
Centre for Digital Music (C4DM),
Queen Mary University of London, UK
g.v.morfi@qmul.ac.uk

Dan Stowell

Machine Listening Lab,
Centre for Digital Music (C4DM),
Queen Mary University of London, UK
dan.stowell@qmul.ac.uk

ABSTRACT

We propose a method to perform audio event detection under the common constraint that only limited training data are available. In training a deep learning system to perform audio event detection, two practical problems arise. Firstly, most datasets are "weakly labelled" having only a list of events present in each recording without any temporal information for training. Secondly, deep neural networks need a very large amount of labelled training data to achieve good quality performance, yet in practice it is difficult to collect enough samples for most classes of interest. In this paper, we propose a data-efficient training of a stacked convolutional and recurrent neural network. This neural network is trained in a multi instance learning setting for which we introduce a new loss function that leads to improved training compared to the usual approaches for weakly supervised learning. We successfully test our approach on two low-resource datasets that lack temporal labels.

Index Terms— Multi instance learning, deep learning, weak labels, audio event detection

1. INTRODUCTION

In recent decades, there has been an increasing amount of audio datasets that have labels assigned to them to indicate the presence or not of a specific event type. This is related to tagging of audio recordings [1, 2, 3, 4]. However, in many cases these labels do not contain any information about the temporal location of each event or the number of occurrences in a recording. This type of label, which we will refer to as *weak*, lacks any temporal information. Collecting and annotating data with strong labels, labels that contain temporal information about the events, is a time consuming task involving a lot of manual labour. On the other hand collecting weakly labelled data takes much less time, since the annotator only has to mark the active sound event classes and not their exact boundaries.

In comparison to supervised techniques trained on strong labels, there has been relatively little work on audio event detection using weakly labelled data. In [5, 6, 7] the authors try to exploit weak labels in birdsong detection and bird species classification, while in [8] singing voice is pinpointed from weakly labelled examples. Furthermore, in [9] the authors train a network that can do automatic scene transcription from weak labels and in [10] audio from YouTube videos is used in order to train and compare different previously proposed convolutional neural network architectures for audio event detection and classification. Finally, in [11, 12] the authors use weakly labelled data for audio event detection in order to move from the weak labels space to strong labels. Most of these methods formulate the provided weak labels of the recordings into a multi instance learning (MIL) problem.

Machine learning has experienced a strong growth in recent years, due to increased dataset sizes and computational power, and to advances in deep learning methods that can learn to make predictions in extremely nonlinear problem settings [13]. However, a large amount of data is needed in order to train a neural network that can achieve a good quality performance. Depending on the audio event to be detected and classified in each task it may become difficult to collect enough samples for them. Annotating data with *strong* labels, labels that contain temporal information about the events, to train audio event detectors is a time consuming process involving a lot of manual labour. On the other hand, collecting weakly labelled data takes much less time, since the annotator only has to mark the active sound event classes and not their exact boundaries. We refer to datasets that only have weak labels, may contain rare events and have limited amounts of training data as *low-resource* datasets.

In this paper, we propose a network that uses low-resource datasets in an efficient way in order to predict audio event detection using only the weak labels provided for each recording during training. This network is trained in a MIL setting where we propose a new loss function that outperforms the most commonly used losses when trying to derive the strong labels from weakly labelled data. The rest of the paper is structured as follows: Section 2 describes the multi instance learning setting, Section 3 presents our method. Evaluations on two different low-resource datasets follow in Section 4, with the conclusions in Section 5.

2. MULTI INSTANCE LEARNING

Training audio event detectors on low-resource datasets presents the issue of weak-to-strong prediction. Low-resource datasets only provide the user with weak labels that don't include any temporal information about the events but only denote the presence or absence of a specific class in a recording. However, audio event detectors produce labels with start and end times, referred to as strong labels, hence provide full temporal information about the events in a recording.

The most common way to train a network for weak-to-strong prediction is the multi instance learning (MIL) setting. The concept of MIL was first properly developed in [14] for drug activity detection. MIL is described in terms of *bags*, with a bag being a collection of instances. The existing weak labels are attached to the bags, rather than the individual instances within them. Positive bags have at least one positive instance, an instance for which the

target class is active. On the other hand, negative bags contain negative instances only. A negative bag is thus pure while a positive bag is presumably impure, since the latter most likely contains both positive and negative instances. There is no direct knowledge of whether an instance in a positive bag is positive or negative. Thus, it is the bag-label pairs and not the instance-label pairs which form the training data, and from which a classifier which classifies individual instances must be learned.

Let the training data be composed of N bags, i.e. $\{B_1, B_2, ..., B_N\}$, the i-th bag is composed of M_i instances, i.e. $\{B_{i1}, B_{i2}, ..., B_{iM_i}\}$, where each instance is a p-dimensional feature vector, e.g. the j-th instance of the i-th bag is $[B_{ij1}, B_{ij2}, ..., B_{ijp}]^T$. We represent the bag-label pairs as (B_i, Y_i), where $Y_i \in \{0, 1\}$ is the bag label for bag B_i. $Y_i = 0$ denotes a negative bag and $Y_i = 1$ denotes a positive bag.

One naïve but commonly used way of inferring the individual instances' labels from the bag labels is assigning the bag label to each instance of that bag: we refer to this method as *false strong labelling*. During training, a neural network in the MIL setting with false strong labels tries to minimise the average divergence between the network output for each instance and the false strong labels assigned to them, identically to an ordinary supervised learning scenario. However, it is evident that the false strong labelling approach is an approximation of the loss for a strong label prediction task, hence it has some disadvantages. When using false strong labels some kind of early stopping is necessary since when minimal loss is achieved that would mean all positive instance predictions for a positive bag. However, there is no clear way of defining a specific point for early stopping. This is an issue that all methods in the MIL setting face.

As an alternative to false strong labels, one can attempt to infer labels of individual instances in bag B_i by making a few educated assumptions. The most common ones are: if $Y_i = 0$, all instances of bag B_i are negative instances, hence $y_{ij} = 0, \forall j$, while on the other hand, if $Y_i = 1$, at least one instance of bag B_i is equal to one. For all instances of bag B_i, this relation between the bag label and instance labels can be simply written as $Y_i = \max_j y_{ij}$. Using this assumption in the MIL setting, we must modify the manner in which the divergence to be minimized is computed, to utilize only weak labels, as proposed in [15].

Let o_{ij} represent the output of the network for input B_{ij}, the j-th instance in B_i, the i-th bag of training instances. We define overall divergence on the training set as the sum of the bag-level divergences E_i, each computed for bag B_i:

$$E = \sum_{i=1}^{N} E_i = \sum_{i=1}^{N} \frac{1}{2} \left(\max_{1 \le j \le M_j} (o_{ij}) - Y_i \right)^2 \quad (1)$$

where Y_i is the label assigned to bag B_i.

This indicates that if at least one instance of a positive bag is perfectly predicted as positive, or all the instances of a negative bag are perfectly predicted as negative, then the error on the concerned bag is zero. Otherwise, the weights will be updated according to the error on the instance whose corresponding actual output is the maximal among all the instances in the bag. Note that such an instance is typically the most easy to be predicted as positive for a positive bag, while it is the most difficult to be predicted as negative for a negative bag. On an instance-level, when using max to compute the loss, only one instance per bag contributes to the gradient, which may lead to inefficient training. In positive bags, the network only has to accurately predict the label for the easiest positive instance to reach a perfect accuracy, thus not paying as much attention to the rest positive instances that might be harder to accurately detect.

In this work, we perform audio event detection by training a neural network using weakly labelled data in a MIL setting. Since the loss function can have a dramatic effect on the utility of MIL training, we propose a loss function at a bag-level that does not have the disadvantages of the above methods and leads to improved training.

3. METHOD

3.1. Loss Function

Using all instances in a bag for computing error and backpropagated gradient is important, since the network ideally should acquire knowledge from every instance in each epoch. However, it is hard to find an elegant theoretical interpretation of the characteristics of the instances in a bag. On the other hand, simple assumptions about these characteristics can achieve a similar effect. One approach is to consider the mean of the instance predictions of a bag. If a bag is negative the mean should be zero, while if it is positive it should be greater than zero. The true mean is unknown in weakly labelled data. A naïve assumption is to presume that approximately half of the time an event will be present in a recording. Even though this is not true all of the time, it takes into consideration the predictions for all instances, and also inserts a bias to the loss that will keep producing some gradient even after the max term has reached its perfect accuracy. Another simple yet accurate assumption is that on both negative and positive recordings the minimum predictions at an instance-level should be zero. It is possible for a positive recording to have no negative frames however it is extremely rare in practice. This assumption could be used in synergy with max and mean to enforce the prediction of negative instances even on positive recordings and manage a certain level of the bias that is introduced with considering mean in the computation of the loss.

Our proposed loss function that takes into account all the above mentioned assumptions is computed as:

$$\begin{aligned} Loss = \frac{1}{3}\big(&bin\_cr(max_j(o_{ij}), Y_i) \\ &+ bin\_cr(mean_j(o_{ij}), \frac{Y_i}{2}) + bin\_cr(min_j(o_{ij}), 0)\big) \end{aligned} \quad (2)$$

where $bin\_cr(x, y)$ is a function that computes the binary cross-entropy between x and y, o_{ij} are all the predicted strong labels of bag B_i, where $j = 1...M_i$ with M_i being the total number of instances in a bag, and Y_i is the label of the bag.

We refer to this as an *MIL setting using MMM*. For negative recordings, (2) will compute the binary cross-entropy between the max, mean and min of the predictions of the instances of a bag B_i and zero. This denotes that the predictions for all instances of a negative recording should be zero. On the other hand, for positive recordings the predictions should span the full dynamic range from zero to one, biased towards a similar amount of positive and negative instances. Our proposed loss function is designed to balance the positive and negative predictions in a bag resulting in a network that has the flexibility of learning from harder-to-predict positive instances even after many epochs. This is due to the fact that there are no obvious local minima to get stuck in as in the max case.

3.2. Training Settings

As input to our proposed method, log mel-band energy is extracted from audio in 23ms Hamming windows with 50% overlap. In order

to do so the `librosa` Python library is used.[1] In total, 40 mel-bands are used in the 0–44100 Hz range. For a given 5 second audio input, the feature extraction produces a 432x40 output ($T = 432$).

We use a stacked convolutional and recurrent neural network architecture (cf. [9]) to predict the strong labels of a recording. Table 1 presents the overall architecture of the proposed method.

Table 1: WHEN network architecture. Size refers to either kernel shape or number of units. #Fmaps is the number of feature maps in the layer. Activation denotes the activation used for the layer

Layer	Size	#Fmaps	Activation
Convolution 2D	3x3	64	Linear
Batch Normalisation	-	-	-
Activation	-	-	ReLU
Max Pooling	1x5	-	-
Convolution 2D	3x3	64	Linear
Batch Normalisation	-	-	-
Activation	-	-	ReLU
Max Pooling	1x4	-	-
Convolution 2D	3x3	64	Linear
Batch Normalisation	-	-	-
Activation	-	-	ReLU
Max Pooling	1x2	-	-
Reshape	-	-	-
Bidirectional GRU	64	-	tanh
Bidirectional GRU	64	-	tanh
Time Distributed Dense	64	-	ReLU
Time Distributed Dense	1	-	Sigmoid
Flatten	-	-	-

The log mel-band energy feature extracted from the audio is fed to our network, which produces the predicted strong labels for each recording. The input to the proposed network is a Tx40 feature matrix. The convolutional layers in the beginning of the network are in charge of learning the local shift-invariant features of this input. The max-pooling operation is performed along the frequency axis after every convolutional layer to reduce the dimension for the feature matrix while preserving the number of frames T. The output of the convolutional part of the network is then fed to bi-directional gated recurrent units (GRUs) to learn the temporal structure of audio events. Next we apply time distributed dense layers to reduce feature-length dimensionality. Note that the time resolution of T frames is maintained in both the GRU and dense layers. A sigmoid activation is used in the last time-distributed dense layer to produce a binary prediction of whether there is an event present in each time frame. This prediction layer outputs a continuous range of [0,1] for each frame of the input features. A prediction value equal or greater than 0.5 is taken to indicate the presence of the audio event in question while a prediction less than 0.5 signifies the absence of it. The dimensions of each prediction are Tx1.

4. EVALUATION

4.1. Datasets

In order to test our approach in a low-resource setting we contacted our experiments on the training dataset provided during the Neural Information Processing Scaled for Bioacoustics (NIPS4B) bird song competition of 2013 and on a subset of the DCASE 2018 dataset used for Task 4: Large-scale weakly labeled semi-supervised sound event detection in domestic environments.[2][3] The first dataset contains birdsong recordings and the second one everyday domestic sound events.

The NIPS4B 2013 training set contains 687 recordings of maximum length of 5 seconds each. The recordings have already been weakly labelled with a total of 87 possible classes. Such a dataset can be considered low-resource for a few reasons. First, the total amount of training time is less than one hour. Also, there are 87 possible labels that have very sparse activations, 7 to 20 positive recordings for each. In order to efficiently use the data provided by the NIPS4B 2013 training dataset we first consider all 87 unique labels as one general label 'bird' and train an audio event detection network for this audio event. Another limitation of this dataset is the imbalance of positive and negative recordings: out of the whole dataset (687 recordings) only 100 of them are labelled as negative. This caused some of our original experiments to classify almost all bags as positive ones. An easy way to solve the positive-negative imbalance is to force the network to have the same amount of positive and negative recordings in each mini-batch. Since the amount of negative recordings is much smaller compared to positive ones, we randomly repeat the negative recordings in each mini-batch. We refer to this as Half and Half (HnH) training. This provides a balanced training set at a bag level, but not necessarily balanced at an instance level.

For the following experiments, we split the NIPS4B 2013 training dataset into a training set and testing set. Only weak labels for the training set during the NIPS4B 2013 bird song competition were released, hence we could only use these recordings. We acquired the strong labels of most of these recordings via manual annotations, in order to use them for evaluating our system and have uploaded our transcriptions online.[4]

In order to evaluate how our method behaves with other types of audio events, we use a subset of the DCASE 2018 dataset of Task 4. In order to create a training set with less than an hour of training data that we can use as a low-resource dataset, we randomly select 360 recordings of maximum length of 10 seconds each out of the 1578 recordings of the task training set. DCASE consists of 10 classes. We combine three of them (Dog, Cat, Speech) into one general category named 'mammals' and use that as our positive class for training our detector. Half and Half training is used in order to balance the amount of negative and positive recordings, that are originally 160 and 200 recordings, respectively. The full testing set of Task 4 is used for evaluating our method.

4.2. False Strong Labelling

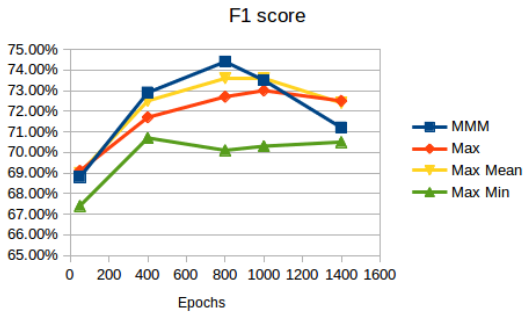
One of the most common ways of training networks from weak to strong labels is generating false strong labels for training by replicating the weak labels for every time frame of the audio and using them as strong labels. We train our network using false strong labels for 3000 epochs using binary cross-entropy as the loss function and Adam optimizer [19]. Figure 2a shows the resulting transcrip-

[1] `https://librosa.github.io/librosa/index.html`

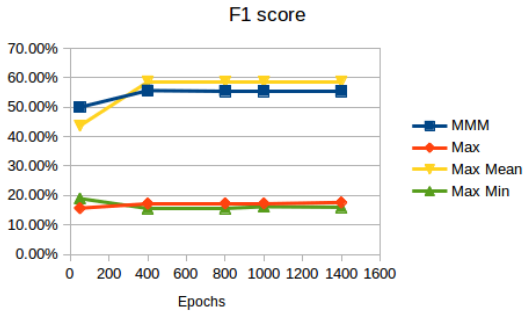
[2] `http://sabiod.univ-tln.fr/nips4b/challenge1.html`

[3] `https://goo.gl/8wKkGk`

[4] `https://figshare.com/articles/Transcriptions_of_NIPS4B_2013_Bird_Challenge_Training_Dataset/6798548`

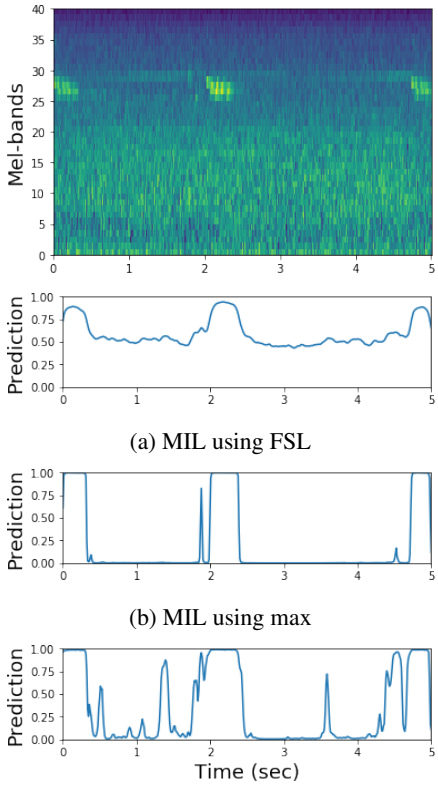


(a) NIPS4B dataset



(b) DCASE dataset

Figure 1: Comparison of the progress of F1 score for our testing sets through epochs for different loss functions.



(a) MIL using FSL

(b) MIL using max

(c) MIL using MMM

Figure 2: Predicted transcription of a recording from the testing set on the NIPS4B dataset. 2a depicts the results of our network trained in a false strong labelling setting. 2b depicts the results of our network trained with max loss. 2c depicts the results of our network trained with MMM loss.

tion predicted by the network. The main issue one can notice is that using false strong labels has a tendency of pushing all the results closer to one when dealing with a positive recording. Some structure is apparent in the transcription, hence the network is indeed able to differentiate between positive and negative instances to some degree, however all results for positive recordings are above the usual 0.5 threshold. We attribute this primarily to the nature of the false strong labels: for a positive recording all time frames are labelled as positive. Furthermore, since perfect accuracy for this setting does not correspond to the actual task, it becomes extremely unclear when training should be stopped.

4.3. MIL using different loss functions

We train two networks using the loss functions described in Section 3, namely max and MMM. Additionally, we trained two more networks using only the max and mean terms of MMM and the max and min terms of MMM to compare their performance and the impact each term has in the predictions of our audio event detector.

Figures 1a and 1b present the progress of the F1 score, the harmonic average of the precision and recall of the predictions, on the NIPS4B and DCASE testing sets respectively, during training. One can notice that methods that use the mean term in the loss prediction tend to reach higher scores. For the NIPS4B dataset, we notice that after training for a certain amount of epochs the results for most methods are decreasing: this is due to the common issue of the MIL setting which is defining when one should stop training.

Another interesting aspect one can study is the individual results of the conventional max loss to our proposed MMM loss. Figure 2 depicts a positive recording from our NIPS4B testing set and the transcriptions predicted by each method. It is evident from these examples that once the max loss reaches the perfect accuracy for a bag, it ignores the harder-to-predict positive events. In this example, the network correctly predicts the three more prominent events and then ignores all other events between them. However, the network trained with a MMM loss is starting to pick out some of the harder to detect events, due to the gradient provided by using mean.

5. CONCLUSIONS

In this paper, we propose a method to perform audio event detection in a MIL setting that introduces a new loss function that takes into account predictions for all instances. Our method is tested on two low-resource datasets with only weakly labelled training data to perform bird and mammal vocalisation detection, respectively. We compare the different components of our loss function and define the influence each of them has to the training of the network. Training a network with our proposed loss outperforms the previously used losses in the MIL setting. Furthermore, our method is not tailored to any specific type of audio events, hence there is reason to believe it can be used for any kind of audio event detection tasks.

6. REFERENCES

[1] Q. Kong, Y. Xu, W. Wang, and M. D. Plumbley, "A joint detection-classification model for audio tagging of weakly labelled data," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, March 2017, pp. 641–645.

[2] Y. Xu, Q. Huang, W. Wang, P. Foster, S. Sigtia, P. J. B. Jackson, and M. D. Plumbley, "Unsupervised feature learning based on deep models for environmental audio tagging," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 25, no. 6, pp. 1230–1241, June 2017.

[3] Y. Xu, Q. Kong, Q. Huang, W. Wang, and M. D. Plumbley, "Convolutional gated recurrent neural network incorporating spatial features for audio tagging," *2017 International Joint Conference on Neural Networks (IJCNN)*, pp. 3461–3466, 2017.

[4] S. Adavanne, K. Drossos, E. Çakir, and T. Virtanen, "Stacked convolutional and recurrent neural networks for bird audio detection," in *2017 25th European Signal Processing Conference (EUSIPCO)*, Aug 2017, pp. 1729–1733.

[5] F. Briggs, B. Lakshminarayanan, L. Neal, X. Fern, R. Raich, S. J. K. Hadley, A. S. Hadley, and M. G. Betts, "Acoustic classification of multiple simultaneous bird species: A multi-instance multi-label approach," *Journal of the Acoustic Society of America*, vol. 131, pp. 4640–4650, 2014.

[6] J. F. Ruiz-Muñoz, M. Orozco-Alzate, and G. Castellanos-Dominguez, "Multiple instance learning-based birdsong classification using unsupervised recording segmentation," in *Proceedings of the 24th International Conference on Artificial Intelligence*, ser. IJCAI'15. AAAI Press, 2015, pp. 2632–2638. [Online]. Available: http://dl.acm.org/citation.cfm?id=2832581.2832617

[7] L. Fanioudakis and I. Potamitis, "Deep networks tag the location of bird vocalisations on audio spectrograms," vol. 1711.04347, 2017.

[8] J. Schlüter, "Learning to Pinpoint Singing Voice from Weakly Labeled Examples," in *Proceedings of the 17th International Society for Music Information Retrieval Conference (ISMIR 2016)*, New York, USA, 2016.

[9] S. Adavanne and T. Virtanen, "Sound event detection using weakly labeled dataset with stacked convolutional and recurrent neural network," in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2017 Workshop (DCASE2017)*, November 2017, pp. 12–16.

[10] S. Hershey, S. Chaudhuri, D. P. W. Ellis, J. F. Gemmeke, A. Jansen, R. C. Moore, M. Plakal, D. Platt, R. A. Saurous, B. Seybold, M. Slaney, R. J. Weiss, and K. Wilson, "Cnn architectures for large-scale audio classification," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, March 2017, pp. 131–135.

[11] A. Kumar and B. Raj, "Audio event detection using weakly labeled data," in *Proceedings of the 2016 ACM on Multimedia Conference*, ser. MM '16. New York, NY, USA: ACM, 2016, pp. 1038–1047. [Online]. Available: http://doi.acm.org/10.1145/2964284.2964310

[12] ——, "Deep CNN framework for audio event recognition using weakly labeled web data," 2017.

[13] Y. Lecun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 5 2015.

[14] T. G. Dietterich, R. H. Lathrop, and T. Lozano-Pérez, "Solving the multiple instance problem with axis-parallel rectangles," *Artificial Intelligence*, vol. 89, no. 1, pp. 31–71, 1997. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0004370296000343

[15] Z.-H. Zhou and M.-L. Zhang, "Neural networks for multi-instance learning," in *Proceedings of the International Conference on Intelligent Information Technology, Beijing, China*, 2002, pp. 455–459.

[16] R. Amar, D. R. Dooly, S. A. Goldman, and Q. Zhang, "Multiple-instance learning of real-valued data," in *Proceedings of the Eighteenth International Conference on Machine Learning*, ser. ICML '01. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2001, pp. 3–10. [Online]. Available: http://dl.acm.org/citation.cfm?id=645530.655815

[17] D. Liu, Y. Zhou, X. Sun, Z. Zha, and W. Zeng, "Adaptive pooling in multi-instance learning for web video annotation," in *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*, Oct 2017, pp. 318–327.

[18] Y. Wang, J. Li, and F. Metze, "Comparing the max and noisy-or pooling functions in multiple instance learning for weakly supervised sequence learning tasks," 2018.

[19] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014.

[20] C. J. V. Rijsbergen, *Information Retrieval*, 2nd ed. Newton, MA, USA: Butterworth-Heinemann, 1979.

APPLYING TRIPLET LOSS TO SIAMESE-STYLE NETWORKS FOR AUDIO SIMILARITY RANKING

Brian Margolis, Madhav Ghei, Bryan Pardo

Northwestern University
Electrical Engineering and Computer Science
2133 Sheridan Rd
Evanston, IL 60208, USA
{brianmargolis, madhavghei2018}@u.northwestern.edu, pardo@northwestern.edu

ABSTRACT

Query by vocal imitation (QBV) systems let users search a library of general non-speech audio files using a vocal imitation of the desired sound as the query. The best existing system for QBV uses a similarity measure between vocal imitations and general audio files that is learned by a two-tower semi-Siamese deep neural network architecture. This approach typically uses pairwise training examples and error measurement. In this work, we show that this pairwise error signal does not correlate well with improved search rankings and instead describe how triplet loss can be used to train a two-tower network designed to work with pairwise loss, resulting in better correlation with search rankings. This approach can be used to train any two-tower architecture using triplet loss. Empirical results on a dataset of vocal imitations and general audio files show that low triplet loss is much better correlated with improved search ranking than low pairwise loss.

Index Terms— vocal imitation, information retrieval, convolutional Siamese-style networks, triplet loss

1. INTRODUCTION

Finding ways to easily access relevant audio content is a task that has increased in importance as multimedia collections proliferate and grow. For example, the widely-used *Freesound*[1] website contains hundreds of thousands of individual sound recordings from many categories of sound. Such repositories typically let users search their collections of recordings using text-based search. This allows search through any tags, descriptions and file names, but does not support search on the content of audio files. This is not true just for online repositories. Sound designers rely on commercially deployed sound library management tools, such as Soundly [1], to index their sound file collections. These systems also search on text-based metadata and not the audio content.

Indexing a collection of audio files using text-based descriptors imposes certain natural limitations on how search may be done. Such descriptions often do not provide the necessary detail to evaluate this sound in comparison to others with a similar label. This forces the user to listen to all sounds sharing a label, which can be prohibitively time-consuming. Relying on text descriptions also means that every file must be assigned text labels before one can search for it. Further, the important fine grain characteristics that differentiate between audio files of the same general category may not have widely agreed upon text descriptors, making it difficult to create tags that support fine grained search.

When words fail, vocal imitations can help to bridge the gap left by text descriptors. Since imitation allows for description of sounds in ways that text cannot [2], using a vocal imitation as the query has the potential to yield useful results when text search fails [3]. Query by vocal imitation (QBV) systems allow search in a collection of sound files using a vocal imitation of the desired sound as the search key.

The current state-of-the-art in QBV [4] measures the similarity of the imitation to each sound in the database using a similarity measure output by a two tower Siamese-style neural network. The network takes a vocal imitation and a sound file from the collection as input and outputs a similarity value in the range [0,1]. These similarity numbers are used to rank audio files in the collection. In training, networks are trained on labeled pairs, where 0 indicates a vocal imitation was paired with the incorrect sound and 1 indicates the imitation was paired with the target of the imitation.

In this work, we show that the error signal provided by this pairwise training does not necessarily correlate well with improving the ranking of the target file. We show that a loss function (triplet loss) that explicitly compares the similarity of the imitation query to two different sounds in the collection is much better correlated with the rank of the target. Finally, we show how to adapt triplet loss training to work in a two-tower architecture (see Section 3). This approach can be used to train any two-tower architecture using triplet loss.

2. RELATED WORK

There are a number of audio search approaches that are related to query by vocal imitation of general sounds. Audio fingerprinting services (e.g. the song-finding service *Shazam*[2]) require the query be a portion of the exact audio file sought. One cannot vocally imitate the desired sound and find a match using audio fingerprinting. There are services that make speech recordings searchable as text (e.g. the Microsoft Speech API [3]), but these are not designed to meaningfully encode general sounds or vocal imitations. Query by humming systems focus specifically on melody (e.g. Tunebot [5])

This work supported by USA National Science Foundation Award 1617497.

[1] https://freesound.org/browse/

[2] https://www.shazam.com

[3] https://docs.microsoft.com/en-us/azure/cognitive-services/speech/home

and rhythm (e.g. Query by Beatboxing [6]) and are not suited for general query by vocal imitation of general sounds.

Synthassist [7] is a search tool for music synthesizer sounds. It compares vocal imitations to a library of synthesizer sounds by creating temporal vectors of standard audio features (e.g. mel-frequency cepstral coefficients) and using an edit-distance to compare the query to audio files in the database. However, in recent years, better retrieval accuracy has been achieved by deep learning models which learn the relevant feature sets during training. The state-of-the-art in query by vocal imitation was improved when convolutional auto encoders were applied to the problem by multiple research groups [8, 9, 10, **?**].

Zhang and Duan further improved upon the QBV accuracy of CAEs with IMINET [11], a two-tower feed-forward convolutional network. In their work, each of the two inputs (an audio file from a database and an imitation to be compared) is encoded by one of the two towers and the output of both towers are input to a fully-connected network that produces a similarity measure between the two audio files. Their most recent work, and the current state of the art in QBV, is TL-IMINET [4], which they call a Siamese-style architecture since the tower that takes the vocal imitation as input has a different architecture than the tower accepting sounds from the collection (as opposed to fully Siamese nets, where the towers share weights and architecture). In all of the recent work, pairwise loss was used in training.

In the domain of image search, Wang et al. [12] proposed a triplet loss method for learning feature models of images where training examples consist of a query and two rank-ordered database elements. This allowed learning fine grained distinctions between images of the same class. They showed this approach outperformed existing models that used hand-crafted features. However they did not compare the triplet loss training approach to pairwise training. They applied their work to a three-tower deep net model but did not show how it could be applied to a two-tower model. Also, their queries (photos) were drawn from the same data as their search results (photos from the same collection as the queries).

Our work combines and builds on ideas from Wang et al.[12] and Zang et al.[4] We illustrate how to adapt an existing two-tower architecture to be trained using triplet loss, instead of forcing the use of an altogether new architecture. We apply this approach to train a system to do pairwise comparison between very different classes of sound objects: vocal imitations and reference audio recordings. An analogous task in the image domain would be to search a set of photos using hand-drawn images as the queries. We then compare the effectiveness of triplet loss to pairwise loss in training a network to solve a ranking problem.

3. METHODS

We assume a set of sound recordings R where r is a recording in the set. The search key is a vocal imitation v of some file in the set, known as the target, t. Given a similarity measure $s(v, r)$ that returns a similarity value in the range [0,1] for any recording, we can provide an ordering of the files in R, based on similarity. This ordering is used as a ranking returned by a search engine. The more consistently the target is returned as a highly ranked recording, the better the search engine. The question then becomes how to create a similarity measure that will consistently rank that target highly.

3.1. TL-IMINET

TL-IMINET [4] is the most successful system, to-date, for QBV. In that work they learn the similarity measure using a neural network with two convolutional towers: one tower for a vocal imitation and the other for a recording from the data set. These towers both learn embeddings that are fed into a fully connected network. A trained network takes a vocal imitation and a reference audio file as input and outputs a value in the range [0,1], where 1 indicates a perfect match.

Our goal is not to design a superior network architecture, but to develop a superior training method. Therefore, we use the exact architecture and audio encoding used in the TL-IMINET paper. We provide an overview of the network structure and audio encoding below. More detail can be found in [4].

The input to the vocal imitation tower is 4 seconds of audio (vocal imitations in the data set are typically less than 4 seconds long). This is encoded as a 39-band mel-spaced spectrogram with 8.33 ms for both the window size and hop size. The resulting input spectrogram has 39 frequency bins and 482 time bins. The vocal imitation tower has three convolutional layers. For the first two convolutional layers, each layer has 48 filters with ReLU activations. Both layers are followed by a 6×6 pooling layer. The third convolutional layer also has 48 filters and a receptive field of 6×6. It is followed by a 1×2 pooling layer with 1 in frequency and 2 in time.

The general audio tower is passed reference recordings truncated to 3 seconds and converted into a mel spectrogram with 23 ms window size and 23 ms hop size. This leads to an input dimensionality of 128 mel-frequency bands by 128 time steps. This is input to a tower with 3 convolutional layers. The first convolutional layer has 24 filters with a receptive field of 5×5, and followed by a ReLU activation function. This is fed into a 2×4 (both shape and stride) max-pooling layer with 2 in frequency and 4 in time. The second and third convolutional layers each have 48 filters with a 5×5 receptive field. The second convolutional layer is followed by a 2×4 pooling layer. Unlike the vocal imitation tower, no pooling layer follows the third convolutional layer.

The embeddings output from each convolutional tower are concatenated and passed into 2 fully connected layers, which calculate a distance between each vector.

3.2. Pairwise loss

TL-IMINET is a Siamese-style network and uses a pairwise loss function. This is the typical loss function used for Siamese and Siamese-style networks. The training pairs consist of a vocal example and a recording from the data set. If the recording in the pair is the target, then the label is 1. If the recording is not the target, the label is 0. The loss function used is binary cross entropy and the goal is for the similarity measure to output 1 for the target of a vocal query and 0 for all other recordings.

While the error signal described above has proven useful in many cases, it may not always correlate strongly with the desired behavior of the system when the goal is to rank order a set of items by similarity to a single query example. For ranking problems, it is not important that the output of the similarity measure be either 1 or 0, for any particular pair. In fact, this goal may even be counterproductive for ranking problems, as we now show. Consider a case with 100 recordings in the data set. Assume the similarity function returned a value in the range [.9, 1] for all recordings and the target is the only recording to get a similarity of 1. The target would be ranked first, which is perfect performance. The error

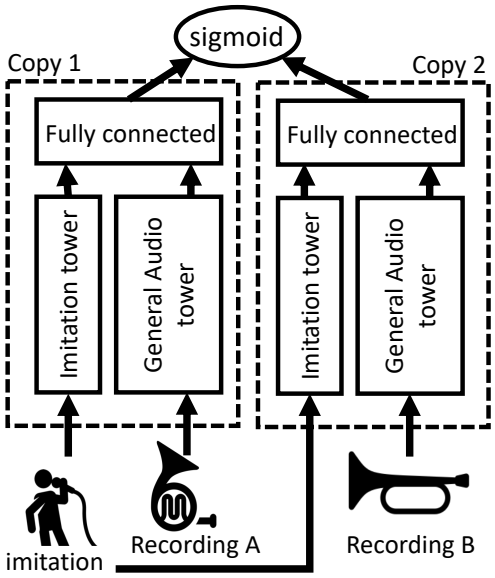


Figure 1: Triplet loss training configuration of the network used to measure similarity between vocal imitations and reference audio files. Two copies of a two-tower Siamese-style network (TL-IMINET) are used. Weights are tied between the two copies. A training example consists of a vocal imitation, the target, and a distractor. The desired output is 1 if the target is recording A and 0 if the target is recording B.

signal, however, would show large amounts of error for 99 out of 100 recordings, since all non-target recordings would be ranked .9 or higher, when 0 is the correct label. Consider another case: the similarity function returns values only in the range $[0, .01]$ and the target is the only file to receive a 0. Here, the error would be very low, since 99 out of 100 recordings were very closed to the training label output of 0, even though the target would be ranked last.

3.3. Triplet loss: an error signal more suited to ranking

In the previous section, we illustrated how pairwise loss may not provide the ideal error signal for ranking problems using two-tower networks. How, then, can the error signal be tied more closely to the desired ranking behavior? In this work, we adopt the idea of *triplet loss*, which has been used successfully to train a three-tower convolutional network to perform fine-grained similarity measurements in the domain of still images [12]. In this paradigm, a training example consists of a triplet (v, a, b). If recording a should be ranked closer to the vocal imitation v, the label is 1. If b should be ranked closer than a, the label is 0. This allows the use of binary cross entropy, but explicitly takes ranking between pairs into account.

We apply this loss function to a Siamese-style network architecture. In our case, that network architecture is TL-IMINET; however, the same approach can be applied to any Siamese or Siamese-style network. Two copies of the Siamese-style architecture are used. Weights are tied between the two copies. The vocal imitation is input to both copies. Recording A is passed to one copy, and recording B is passed to the other copy. The output of both copies is passed to a single sigmoid node that outputs a value in the range 0 to 1. This is illustrated in Figure 1.

The two input weights of the sigmoid are fixed to be +50 and -50, with a bias of 0. This yields the function $\sigma(50a - 50b)$. With this configuration, the output tends to 1 if the output from recording A is higher than B and tends towards 0 if B is higher than A. This configuration allows for triplet-loss training. [4]

Once trained, either copy of the TL-IMINET architecture can be extracted and used in testing to estimate similarity between a vocal imitation query and a recording in the database. Recordings can be ranked by similarity as is done using the two tower TL-IMINET.

By training in this fashion we can apply a triplet loss approach to train a two-tower network designed for pairwise loss. This also allows for a truly meaningful comparison between triplet and pairwise loss, since both training approaches modify the exact same number of weights, and the testing architecture is identical for both approaches.

4. EXPERIMENTAL DESIGN

We have argued that using triplet loss will result in a error signal that is more correlated with the goal (ranking the right answer highly) than happens with the pairwise error signal used in previous vocal imitation search work. We tested this hypothesis by measuring the correlation between improving on the loss function and improving search results as training progresses. To ensure a controlled experiment we used a TL-IMINET architecture, trained using pairwise training and compare the results to an identical TL-IMINET with the same initialization weights, trained using triplet loss. We repeated the comparison on a variety of data splits and with a variety of initializations and measured the statistical difference between the two training approaches. We now describe the experiment in detail.

4.1. Data Set

For this work, we used the Vocal Imitation Set [13], a collection of crowd-sourced vocal imitations of a set of 302 classes of sound. The classes were drawn from Google's AudioSet ontology [14]. Each sound class has an average of 10 clean, single-sound recordings taken from FreeSound (e.g. 10 police siren recordings). A single one of the 10 recordings in each class was used as a reference recording (the *target*) for the vocal imitations. Given a reference recording as the target, Amazon Mechanical Turk workers were asked to record a vocal imitation of the target. Recorded imitations were evaluated by expert listeners and only those recordings that were of sufficiently high quality were used. This resulted in 5,601 high quality imitations of 302 sound classes, or roughly 19 imitations per sound class. For more detail on this data set, please see [13].

4.2. A single trial

A recording of a vocal imitation of a sound is the *query*. The *target* is the single audio file that the query is an imitation of. A *distractor* is a file in the collection that is not the search goal.

In a single trial we randomly select 10 sound classes from the vetted Vocal Imitation data set of 302 sound classes. Each sound class contains an average of 19 imitations and 10 reference recordings. This results in a set of roughly 100 reference files and 190

[4] Note the value of 50 is simply selected to be sufficiently large to saturate the sigmoid activation function, and is not a magic number.

imitations. Training examples for the pairwise loss function require one imitation and one reference (either the target file, or some other file), resulting roughly 19,000 unique training pairs, of which 190 are positive pairs and 18810 are negative. Each training example for the triplet loss function requires one imitation and two reference recordings (the target + distractor), also resulting in roughly 1,881,000 unique training triplets. This data is split into validation (30%) and training (70%) data.

A coarse grained comparison in triplet loss is one where the distractor is drawn from a different sound class than the target. In the case of triplet loss, there are many more coarse grained examples than fine grained examples. Balancing coarse (across class) and fine (within class) distinctions is desirable. Therefore, on each epoch, we train using all the fine grained examples and an equally sized, randomly selected subset of the coarse grained examples. In the case of pairwise lose, there are many more negative examples than positive examples. Similarly, on each epoch, we train using all the positive examples and an equally sized, randomly selected subset of the negative examples.

For each trial, we randomly initialized a TL-IMINET network. We trained the network twice from the same initialization weights, once with triplet loss and once with pairwise loss. We used the ADAM optimization function [15]. See Section 3 for details of the loss functions. Each network was trained for 300 epochs. At each epoch, we use the trained network as a similarity measure to rank the target for each of the vocal imitations among the 100 reference files. The mean rank of the target, as well as the loss function, is recorded at each epoch for both the training and validation data.

The code used to run these trials can be found at our Github repository[5].

5. RESULTS

We ran 28 trials and measured Pearson's correlation coefficient between the ranking results and loss curves for both loss function over the 300 training epochs in each trial. See Figure 2 for loss and rank curves for a representative trial and their correlations. It is clear that triplet loss correlates much better with ranking results than pairwise loss does. This trial was chosen because its correlation between the loss function and ranking results was close to the mean correlation over all the trials, for both pairwise and triplet loss.

We performed a Wilcoxon signed-rank test on the 28 trials, comparing the Pearson correlation of triplet loss to ranking results with the Pearson correlation of pairwise loss to ranking results. The resulting p-value of 5.3×10^{-6}, indicates the improvement in correlation between rank and loss gained from switching to a triplet loss function is statistically significant. Figure 3 shows the distribution of the correlation.

6. CONCLUSIONS

We have shown how, with a simple modification to training, any two tower Siamese-style network can be adapted to learn using triplet loss. We have shown that, for QBV on a dataset of audio files, triplet loss correlates much more closely with ranking results than pairwise loss does. This higher correlation means that the network learns to perform a task closely related to the end-goal of search. This approach to training is promising in that it can be easily applied to any existing Siamese-style network.

[5] https://git.io/fNMfe.

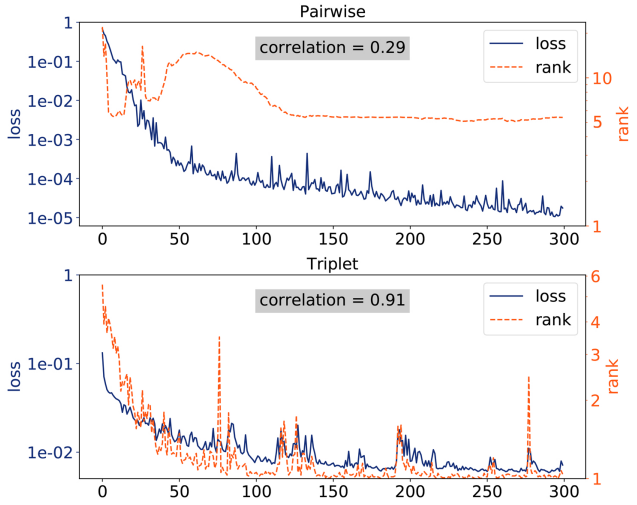


Figure 2: One representative trial. Training loss as a function of training epoch is shown in blue. We measured mean rank across 133 queries in a 100-file search on the training set. This curve is shown in orange. Lower ranking is better. The upper panel shows traditional pairwise loss. The lower shows triplet loss. Lower loss is better. Correlation is the Pearson correlation between the loss and the target rank.

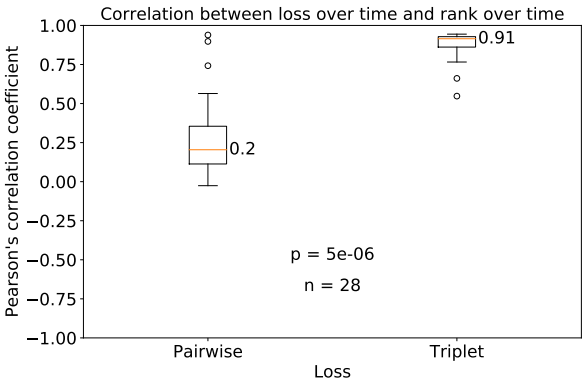


Figure 3: Distribution of the correlation between the search rank of the target file and the the loss function used in training. The value for each trial is the Pearson correlation coefficient between the loss function and the rank of the target. Higher correlation is better. Numbers next to boxes are median values.

7. REFERENCES

[1] "The complete sfx platform." [Online]. Available: https://www.getsoundly.com/

[2] G. Lemaitre and D. Rocchesso, "On the effectiveness of vocal imitations and verbal descriptions of sounds," *The journal of the Acoustical Society of America*, vol. 135, no. 2, pp. 862–873, 2014.

[3] G. Lemaitre, O. Houix, F. Voisin, N. Misdariis, and P. Susini,

"Vocal imitations of non-vocal sounds," *PloS one*, vol. 11, no. 12, p. e0168167, 2016.

[4] Y. Zhang and Z. Duan, "Visualization and interpretation of siamese style convolutional neural networks for sound search by vocal imitation (to be presented)," in *Acoustics, Speech and Signal Processing (ICASSP), 2018 IEEE International Conference on*. IEEE, 2018.

[5] A. Huq, M. Cartwright, and B. Pardo, "Crowdsourcing a real-world on-line query by humming system," in *Proceedings of the Sixth Sound and Music Computing Conference (SMC 2010)*, 2010.

[6] A. Kapur, M. Benning, and G. Tzanetakis, "Query-by-beat-boxing: Music retrieval for the dj," in *Proceedings of the International Conference on Music Information Retrieval*, 2004, pp. 170–177.

[7] M. Cartwright and B. Pardo, "Synthassist: Querying an audio synthesizer by vocal imitation." in *NIME*. Citeseer, 2014, pp. 363–366.

[8] Y. Zhang and Z. Duan, "Imisound: An unsupervised system for sound query by vocal imitation," in *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*. IEEE, 2016, pp. 2269–2273.

[9] ——, "Retrieving sounds by vocal imitation recognition," in *Machine Learning for Signal Processing (MLSP), 2015 IEEE 25th International Workshop on*. IEEE, 2015, pp. 1–6.

[10] A. Mehrabi, K. Choi, S. Dixon, and M. Sandler, "Similarity measures for vocal-based drum sample retrieval using deep convolutional auto-encoders," *arXiv preprint arXiv:1802.05178*, 2018.

[11] Y. Zhang and Z. Duan, "Iminet: Convolutional semi-siamese networks for sound search by vocal imitation," in *Applications of Signal Processing to Audio and Acoustics (WASPAA), 2017 IEEE Workshop on*. IEEE, 2017, pp. 304–308.

[12] J. Wang, Y. Song, T. Leung, C. Rosenberg, J. Wang, J. Philbin, B. Chen, and Y. Wu, "Learning fine-grained image similarity with deep ranking," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 1386–1393.

[13] B. Kim, M. Ghei, B. Pardo, and Z. Duan, "Vocal imitation set: a dataset of vocally imitated sound events using the audioset ontology," in *Proceedings of the 2018 Workshop on Detection and Classification of Acoustic Scenes and Events (DCASE2018)*, 2018.

[14] J. F. Gemmeke, D. P. Ellis, D. Freedman, A. Jansen, W. Lawrence, R. C. Moore, M. Plakal, and M. Ritter, "Audio set: An ontology and human-labeled dataset for audio events," in *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*. IEEE, 2017, pp. 776–780.

[15] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

TO BEE OR NOT TO BEE: INVESTIGATING MACHINE LEARNING APPROACHES FOR BEEHIVE SOUND RECOGNITION

Inês Nolasco and Emmanouil Benetos

School of Electronic Engineering and Computer Science, Queen Mary University of London, UK
{i.nolasco@se17.,emmanouil.benetos@}qmul.ac.uk

ABSTRACT

In this work, we aim to explore the potential of machine learning methods to the problem of beehive sound recognition. A major contribution of this work is the creation and release of annotations for a selection of beehive recordings. By experimenting with both support vector machines and convolutional neural networks, we explore important aspects to be considered in the development of beehive sound recognition systems using machine learning approaches.

***Index Terms*—** Computational bioacoustic scene analysis, ecoacoustics, beehive sound recognition.

1. INTRODUCTION

A significant part of computational sound scene analysis research involves the development of methods for automatic analysis of sounds in natural environments. This area of research has close links with the field of bioacoustics and has several applications, including automatic biodiversity assessment and automatic animal welfare monitoring [1]. Within the context of *computational bioacoustic scene analysis*, the development of technologies for automated beehive monitoring has the potential to revolutionise the beekeeping profession, with benefits including but not limited to a reduction of manual inspections, distant monitoring of bee populations, and by rapidly identifying phenomena related to the natural cycle of the beehive (e.g. queen missing, bee swarming).

In particular, sound plays a central role towards the development of such technologies for automated beehive monitoring. In [2, 3], the authors give a thorough description of bee sounds and their characteristics. In short, the sound of a beehive is a mixture of the individual contributions of sounds produced by each bee of the colony. This mixture is perceived as a dense, continuous, low-frequency buzz.

The first step towards the creation of audio-based beehive monitoring technologies is to create systems that are able to recognise bee sounds and discriminate them from other sounds that might be captured. These non-bee sounds will usually be related with the environment and events occurring in the hive's surroundings and can be as varied as urban sounds, animals, rain, or maintenance sounds. Thus, the aim of this work is to automatically detect sounds produced by bees, distinguishing them from external non-related sounds, given audio recordings captured inside beehives. One aspect that appears useful to differentiate between both classes is that the majority of non-beehive sounds can be of a short duration when compared with beehive sounds.

This work was supported by UK EPSRC grant EP/R01891X/1 and a UK RAEng Research Fellowship (RF/128).

Related works in beehive sound analysis generally use heavy data pre-processing, hand-crafted features and domain knowledge to clean the recordings and come up with useful representations for beehive audio signals. In [4], the authors apply at a first stage a Butterworth filter with cut-off frequencies of 100 Hz and 2000 Hz in order to filter the acoustic signal and remove all sounds of frequencies expected not to be in the bee sound class. In [5], besides the use of several filtering techniques, the authors propose the use of Mel-frequency cepstral coefficients (MFCCs) as features to represent beehive sounds, inspired by speech processing research. The work of [6] is directly relevant to this paper, since a classification is performed to clean the recordings from external sounds. This task is set up to distinguish between 3 classes: beehive sounds, environmental sounds and cricket sounds. However, denoising techniques and hand-crafted features are still applied, including Wavelet transforms and features such as MFCCs, chroma and spectral contrast.

Machine learning methods, and in particular deep learning methods, can decrease up to a point the amount of handcrafted features and domain knowledge which can be responsible for introducing bias and limiting the modelling capabilities of sound recognition methods. In [7], deep neural networks (DNNs) and convolutional neural networks (CNNs) are used to automatically detect the presence of mosquitoes in a noisy environment, although the proposed methodology disregards the long duration characteristics of mosquito sounds. The work of [8] tackles the problem of detecting the presence of birds from audio as part of the 2017 Bird Audio Detection challenge[1]. The proposed method, *Bulbul*, is a combination of deep learning methods also relying on data augmentation. Given that *Bulbul* was the challenge submission that produced the best results, it became the baseline method for the DCASE 2018 Bird Audio Detection task[2]. In the context of environmental sound scene analysis, it is shown in [9] that DNNs have good performance when compared to shallower methods such as Gaussian mixture models (GMMs). However the authors also stress that the use of temporal methods such as recurrent neural networks (RNNs) does not improve classification in this context, which they justify with the characteristic of environmental sounds as not having strong temporal dependencies and being rather non-predictive and random.

In this work, we aim to explore the potential of machine learning methods to the problem of beehive sound recognition, as a first step towards the creation of audio-based beehive monitoring systems. A core problem when using supervised machine learning methods is the large amount of labelled data needed. A major contribution of this work is the creation and release of annotations for a

[1] http://machine-listening.eecs.qmul.ac.uk/bird-audio-detection-challenge/

[2] http://dcase.community/challenge2018/task-bird-audio-detection

selection of recordings from the Open Source Beehive project [10] and for a part of the NU-Hive project dataset [11]. The annotated data is used in experiments using support vector machines (SVMs) and a CNN-based approach by adapting the *Bulbul* implementation [8]. The results presented are indicative of the important aspects to be considered in the development of machine learning-based beehive sound recognition systems.

The outline of the paper is as follows. In Section 2 we describe the data and the annotation procedure. Section 3 describes the methods applied; Section 4 presents the experiments performed, the evaluation metrics, and results. Finally, Section 5 concludes the paper and provides directions for future research.

2. DATA ANNOTATION

The main issue of posing the problem of automatic recognition of beehive sounds as a classification problem is the need for annotated data. In this case we need examples of pure beehive sounds and examples of external sounds as they occur in the recordings made inside the beehives, so that the methods can learn their characteristics and map them to the corresponding labels. Given the lack of labelled data for this task, a major effort of developing such a dataset is undertaken here. The resulting dataset is based on a selected set of recordings acquired in the context of two projects: the Open Source Beehive (OSBH) project [10] and the NU-Hive project [11]. The main goal of both projects is to develop beehive monitoring systems capable of identifying and predicting certain events and states of the hive that are of interest to beekeepers. Among many different variables that can be measured and that help the recognition of different states of the hive, the analysis and use of the sound the bees produce is a big focus for both projects.

The recordings from the OSBH project [10] were acquired through a citizen science initiative which asked members of the general public to record the sound from their beehives together with the registering of the hive state at the moment. Because of the amateur and collaborative nature of this project, the recordings from the OSBH project present great diversity due to the very different conditions in which the signals were acquired: different recording devices used, different environments where the hives were placed, and even different position for the microphones inside the hive. This variety of settings makes this dataset a very interesting tool to help evaluate and challenge the methods developed.

The NU-Hive project [11] is a comprehensive effort of data acquisition, concerning not only sound, but a vast amount of variables that will allow the study of bee behaviours. Contrary to the OSBH project recordings, the recordings from the NU-Hive project are from a much more controlled and homogeneous environment. Here the occurring external sounds are mainly traffic, honks and birds.

The annotation procedure consists in listening the selected recordings and marking the onset and offset of every sound that could not be recognised as a beehive sound. The recognition of external sounds is based primarily on the perceived heard sounds, but a visual aid is also used by visualising the log-mel-frequency spectrum of the signal. All the above are functionalities offered by Sonic Visualiser[3], which was used by two volunteers that are neither bee-specialists nor specially trained in sound annotation tasks. By marking these pairs of instances corresponding to the beginning and end of external sound periods, we are able to get the whole recording labelled into *Bee* and *noBee* intervals. The *noBee* intervals refer to periods where an external sound can be perceived (superimposed to the bee sounds). An example of this process is shown in Fig. 1.

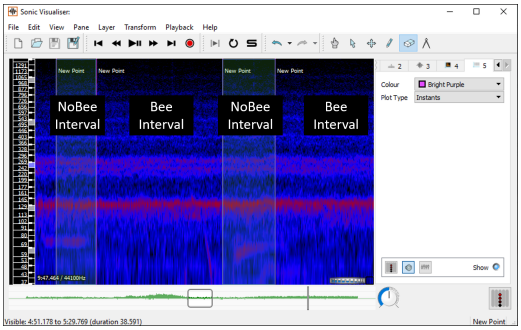


Figure 1: Example of the annotation procedure for one audio file.

The whole annotated dataset consists of 78 recordings of varying lengths which make up for a total duration of approximately 12 hours of which 25% is annotated as noBee events. About 60% of the recordings are from the NU-Hive dataset and represent 2 hives, the remaining are recordings from the OSBH dataset and 6 different hives. The recorded hives are from 3 regions: North America, Australia and Europe. The annotated dataset[4] and auxiliary Python code[5] are publicly available.

3. METHODS

3.1. Preprocessing

The audio recordings are processed at a 22050 Hz sample rate, and are segmented in blocks of predefined lengths. Segments smaller than the defined block length have their length normalised by repeating the audio signal until the block length is reached. For each block a label is assigned based on the existing annotations. A label *Bee* is assigned if the entirety of the segment does not contain nor overlap any external sound interval. Similarly, the label *noBee* is assigned if at least a part of the segment contains an external sound event. Finally, the training data is artificially balanced by randomly duplicating segments of the class less represented.

In order to evaluate the impact of the length of external sounds, we explore different threshold values (Θ) for the minimum duration of external sounds to be included in the annotations.

3.2. SVM classifier

We first create a system for beehive sound recognition using a support vector machine (SVM) classifier. In order to gain insight on which features, normalisation strategies and other classifier parameters are promising to use in this problem, we explore a set of combinations of the three on the SVM classifier, detailed in Section 4.3. Two types of features are extracted for use with the SVM: 20 Mel-frequency cepstral coefficients (MFCCs) and Mel spectra [12], the latter with 80 and 64 number of bands. The spectra are computed with a window size of 2048 samples and hop size of 512 samples.

[3] http://sonicvisualiser.org/

[4] https://zenodo.org/record/1321278#.W2XswdJKjIU

[5] https://github.com/madzimia/Audio_based_identification_beehive_states

3.3. CNN classifier

For the deep learning approach we explore the application of the *Bulbul* CNN implementation [8] as modified for the DCASE 2018 Bird Audio Detection task. The choice of this implementation for a first experiment using a deep learning approach is due to both its promising results achieved in the Bird Audio Detection Challenge, but also because the original problem for which the *Bulbul* system was developed poses similar challenges as the ones we face.

In this implementation, Mel spectra with 80 bands are computed using a window size of 1024 samples and a hop size of 315 samples. Additionally, these spectra are normalised by subtracting their mean over time. The network consists of four convolution layers (two layers of 16 filters of size 3×3 and two layers of 16 filters of size 3×1) with pooling, followed by three dense layers (256 units, 32 units and 1 unit). All layers use a leaky rectifier as activation function with the exception of the output layer which uses the sigmoid function.

Data augmentation is also employed, which includes shifting the training examples periodically in time, and applying random pitch shifting of up to 1 mel band. Dropout of 50% is applied to the last three layers during training.

4. EVALUATION

4.1. Experimental setup

Given the diversity of the data available we are interested in evaluating how well the classifiers are able to generalise to different data. Thus, besides random splitting between train and test sets, we implement a "hive-independent" splitting scheme. This means having training samples belonging only to certain hives, and testing using samples from other, unseen hives.

For both schemes a test size of 5% is used (5% of the total number of segments in the case of the random split scheme or 5% of the number of hives in the hive-independent splitting scheme). When applying the SVM classifier, all remaining data is used in a single training set. For the *bulbul* implementation, in order to mimic the original cross validation scheme, where a model is trained in each set and validated on the others, the remaining data (95%) is further split in half between two sets.

The training of the *Bulbul* network is done by stochastic gradient descent optimisation on a mini-batch of 20 input samples of size 1000 frames by 80 Mel-frequencies (receptive field), and through 100 epochs. The training samples are organised in two sets, and the resulting two trained models are ensembled to generate the predictions in the test set. The prediction for a single sample is obtained by averaging the network output predictions of the non-overlapping 1000 frame excerpts that constitute the whole input sample.

4.2. Evaluation Metrics

The results of each experiment are evaluated using the area under the curve score (AUC) [13]. Each experiment is run three times following the same setup and parameters, and we report the results on each run and the average of the three. The results on the training set are also reported.

4.3. SVM Experiments

As mentioned in Section 3, in this approach a combination of the below parameters is evaluated:

SVM kernels: RBF, linear, and 3rd order polynomial.

Features: μ and σ of: 20 MFCCs, the Δ of 20 MFCCs and of the $\Delta\Delta$ of 20 MFCCs; μ and σ of: Mel-spectra and Δ of Mel-spectra with 64 or 80 bands; μ and σ of: log Mel-spectra and Δ of log Mel-spectra with 64 or 80 bands;

Normalisation strategies: no normalisation, normalisation by maximum value per recording, by maximum value in dataset, z-score normalisation at recording level, and z-score normalisation at dataset level.

Segment size (S): 30 seconds and 60 seconds.

Threshold Θ: 0 seconds and 5 seconds.

Split modes: Hive-independent and Random split

Combining these parameters and evaluating the results of each combination leads us to define the optimal set of parameters (C*). In order to thoroughly evaluate the classifier, experiments using C* are compared against specific parameter changes: (a) different value of threshold Θ; (b) different segment size S; (c) Hive-independent split of the data to determine the generalisation capability to unseen hives; (d) Unbalanced dataset to determine the robustness of the classifier regarding unbalanced classes.

4.4. CNN Experiments

Where possible, parallel experiments to the SVM approach are set up here. As baseline parameters (B*), we use the following:

Features: 80 Mel-band spectra

Receptive field: 1000 frames

Number of training epochs: 100

Batch size: 20

Experiments with changes to these parameters are: (a) different values of Θ, to determine if the classifier can learn to reject only external sounds with long durations; (b) different values of segment size S; (c) Hive-independent split of data, to determine the generalisation capability of the classifier to unseen hives; (d) unbalanced dataset, to determine how the classifier can cope with this aspect; (e) larger receptive fields, to determine if the classifier can exploit the larger context of the input samples.

4.5. SVM Results

The resulting average AUC scores for the test and training set of the 3 runs of each experiment are shown in Fig. 2. From the 1st experiment we infer that the highest average AUC score in test sets is achieved when we use the following combination of parameters (C*): features as the μ and σ of the value, the Δ and the $\Delta\Delta$ of 20 MFCCs, not considering the first coefficient; S of 60 seconds, Θ of 5 seconds and not using any of the normalisation strategies defined.

Fig. 2 **[Θ: 0sec]** shows the AUC results for the experiment using the C* parameters but changing Θ from 5 to 0 seconds. These show primarily that the classifier is not performing in a consistent way, which may indicate a strong dependency on the individual instances in which it is being tested and trained. Also the larger difference between the scores in the train and test sets indicate overfitting to the training examples. Using the smallest value for Θ means that we provide to the classifier samples from which their label is defined based on what can be very short duration events. It is therefore expected that the classifier struggles to distinguish the classes.

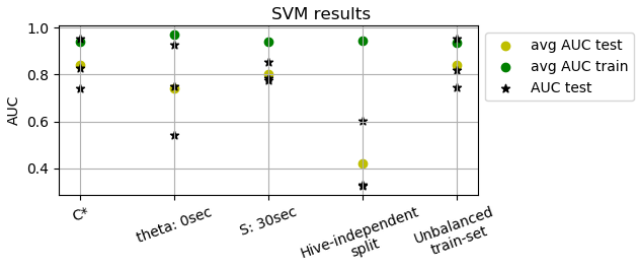


Figure 2: SVM results on the test set for each of the 3 runs (⋆), using the AUC score. The ● and ● represent the average AUC score of the 3 runs in both train and test sets respectively.

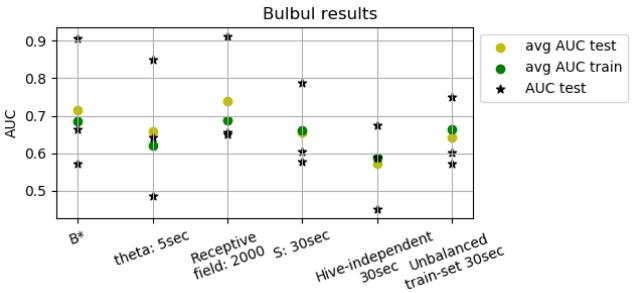


Figure 3: Results for the *Bulbul* CNN using the AUC score, for each of the 3 runs (⋆). The ● and ● represent the average AUC score of the 3 runs in both train and test sets respectively.

By running the classifier with C* parameters but with segment size changed from 60 to 30 seconds (Fig. 2 **[*S*: 30sec]**), we can observe a decrease in both AUC in the train and test sets. These results affirm the idea that, given the long-term aspect of the beehive sounds, if we provide more context to the classifier, it will be better at distinguishing between the two classes of sounds.

In Fig. 2 **[Hive-independent split]**, the classifier is run on 3 sets of data split using the hive-independent splitting scheme. The results clearly show the inability of the classifier to generalise to unseen hives.

Fig. 2 **[Unbalanced train-set]** shows the results of running the classifier in the same sets as experiment C*, but not replicating samples to artificially balance the sets. Comparing the two, they are almost identical which makes sense for SVMs since when data balancing is performed by simple data duplication, the new points are all in locations where data points already existed, therefore these do not influence the decision boundary found by the SVM.

4.6. CNN Results

The resulting average AUC scores for the test and training sets for the 3 runs of each experiment are shown in Fig. 3. The first experiment determined that the best average AUC in the test sets of the 3 runs is achieved when we use the baseline parameters defined in 4.4 plus the following parameters: S of 60 seconds and Θ of 0 seconds. The best results are shown in Fig. 3 **[B*]**.

Regarding the values of Θ, Fig. 3 **[Θ: 5sec]** shows that using a larger Θ is detrimental to performance. This may be explained by the fact that the *Bulbul* system was specifically designed for the detection of bird sounds, which are mainly short duration events, and thus struggles to identify longer events like traffic and rain sounds.

The experiment to evaluate if providing more context to the network improves performance is done by changing the receptive field from 1000 (∼14 seconds) to 2000 (∼30 seconds). In Fig. 3 **[Receptive field: 2000]**, the results show that indeed more context is particularly useful in the context of this problem. This is also consistent with the results from the SVM approach.

The role of S in the CNN approach is different from the SVM one. Here, a larger segment size does not imply that larger samples with more context are given to the classifier, since this is controlled by the receptive field of the network. However, given that prediction is done for a whole segment by averaging the predictions for each frame, using larger segments leads to introducing more context. Confirming the results regarding the need for more context, Fig. 3 **[S: 30sec]** shows that using a smaller segment size results in slightly worse predictions than using a larger segment size (S: 60 seconds, shown in Fig. 3 **[B*]**).

Fig. 3 **[Hive-independent 30sec]** shows the results when using a hive-independent splitting scheme in a 30 second segment size data. Comparing this with the results in Fig. 3 **[S: 30sec]**, the lack of generalisation capacity to unseen hives is also evident here, although, compared with the SVM approach, the results seem to be slightly better and less overfitting occurs which may indicate better generalisation capabilities for the CNN.

Fig. 3 **[Unbalanced train-set 30sec]** shows the results of not doing data balancing on the 30 second segment data. When comparing with Fig. 3 **[S: 30sec]**, the results indicate that data balancing should be considered when training this CNN.

5. CONCLUSIONS

In this work we allocate a major effort for the creation of an annotated dataset for beehive sound recognition where machine learning approaches can be used. However, the annotation procedure can be improved for future additions to this dataset: ideally annotations should be performed by specialists which label overlapping sets of data so that the annotations are subject to peer validation. Finally the main critique to the annotations could be that they are the most important source of human bias introduced in this work.

Although the scores achieved by the CNN implementation fail to achieve the level of the SVM approach, results are indicative of the important aspects to be considered when developing neural networks to tackle this unique problem. Mainly, the importance of providing samples with large context, the amount of training data, and finally due to the incapacity of both approaches to generalise to different hives, the one constraint would be to train systems in the same hives where they are going to be used. We consider that this work can be a first step in a pipeline of beehive monitoring systems, which we think will have an important role in the future of bee keeping. Finally, we expect that this work and the release of the annotated dataset to further motivate research in this topic, and more broadly in the intersection of machine learning and bioacoustics.

6. ACKNOWLEDGEMENT

We would like to thank the authors of the NU-Hive project for creating such complete dataset and making it available for us to work with. Also a special thanks to Ermelinda Almeida for her effort and dedication on annotating the data.

7. REFERENCES

[1] D. Stowell, "Computational bioacoustic scene analysis," in *Computational Analysis of Sound Scenes and Events*, T. Virtanen, M. D. Plumbley, and D. P. W. Ellis, Eds. Springer, 2018, pp. 303–333.

[2] M. Bencsik, J. Bencsik, M. Baxter, A. Lucian, J. Romieu, and M. Millet, "Identification of the honey bee swarming process by analysing the time course of hive vibrations," *Computers and Electronics in Agriculture*, vol. 76, no. 1, pp. 44–50, 2011.

[3] A. Zacepins, A. Kviesis, and E. Stalidzans, "Remote detection of the swarming of honey bee colonies by single-point temperature monitoring," *Biosystems Engineering*, vol. 148, pp. 76–80, 2016.

[4] S. Ferrari, M. Silva, M. Guarino, and D. Berckmans, "Monitoring of swarming sounds in bee hives for early detection of the swarming period," *Computers and Electronics in Agriculture*, vol. 64, no. 1, pp. 72–77, 2008.

[5] A. Robles-Guerrero and T. Saucedo-Anaya, "Frequency analysis of honey bee buzz for automatic recognition of health status: A preliminary study," *Research in Computing Science*, vol. 142, no. 2017, pp. 89–98, 1870.

[6] P. Amlathe, "Standard machine learning techniques in audio beehive monitoring: Classification of audio samples with logistic regression, K-nearest neighbor, random forest and support vector machine," Master's thesis, Utah State University, 2018.

[7] I. Kiskin, P. Bernardo, T. Windebank, D. Zilli, and M. L. May, "Mosquito detection with neural networks: The buzz of deep learning," *ArXiv e-prints*, pp. 1–16, arXiv:1705.05180v1.

[8] T. Grill and J. Schlüter, "Two convolutional neural networks for bird detection in audio signals," in *25th European Signal Processing Conference (EUSIPCO)*, 2017, pp. 1764–1768.

[9] J. Li, W. Dai, F. Metze, S. Qu, and S. Das, "A comparison of deep learning methods for environmental sound detection," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Mar. 2017, pp. 126–130.

[10] "Open Source Beehives Project," https://www.osbeehives.com/.

[11] S. Cecchi, A. Terenzi, S. Orcioni, P. Riolo, S. Ruschioni, and N. Isidoro, "A preliminary study of sounds emitted by honey bees in a beehive," in *Audio Engineering Society Convention 144*, 2018.

[12] R. Serizel, V. Bisot, S. Essid, and G. Richard, "Acoustic features for environmental sound analysis," in *Computational Analysis of Sound Scenes and Events*, T. Virtanen, M. D. Plumbley, and D. P. W. Ellis, Eds. Springer, 2018, pp. 13–40.

[13] A. Mesaros, T. Heittola, and D.Ellis, "Datasets and evaluation," in *Computational Analysis of Sound Scenes and Events*, T. Virtanen, M. D. Plumbley, and D. P. W. Ellis, Eds. Springer, 2018, pp. 13–40.

UNSUPERVISED ADVERSARIAL DOMAIN ADAPTATION FOR ACOUSTIC SCENE CLASSIFICATION

Shayan Gharib[1*], *Konstantinos Drossos*[1*], *Emre Çakir*[1], *Dmitriy Serdyuk*[2], *and Tuomas Virtanen*[1]

[1]Audio Research Group, Lab. of Signal Processing,
Tampere University of Technology, Tampere, Finland
[2]Montreal Institute for Learning Algorithms, Montreal, Canada

ABSTRACT

A general problem in acoustic scene classification task is the mismatched conditions between training and testing data, which significantly reduces the performance of the developed methods on classification accuracy. As a countermeasure, we present the first method of unsupervised adversarial domain adaptation for acoustic scene classification. We employ a model pre-trained on data from one set of conditions and by using data from other set of conditions, we adapt the model in order that its output cannot be used for classifying the set of conditions that input data belong to. We use a freely available dataset from the DCASE 2018 challenge Task 1, subtask B, that contains data from mismatched recording devices. We consider the scenario where the annotations are available for the data recorded from one device, but not for the rest. Our results show that with our model agnostic method we can achieve $\sim 10\%$ increase at the accuracy on an unseen and unlabeled dataset, while keeping almost the same performance on the labeled dataset.

Index Terms— Adversarial domain adaptation, acoustic scene classification

1. INTRODUCTION

The task of acoustic scene classification is to assign to a sound segment the acoustic scene that it belongs to, e.g. office, park, tram, etc. Recently proposed methods for acoustic scene classification (ASC) are based on deep neural networks (DNNs)[1, 2]. They usually employ convolutional neural networks (CNNs) to extract discriminative features from the used data, then using these features as an input to a classifier for classifying the acoustic scene [3, 4, 5, 6]. In a realistic scenario, a method for ASC will be used to classify data emerging from a variety of different domains (i.e. acoustic conditions, acoustic channels) from the data used for optimizing that particular method. The mismatched domains introduce the dataset bias (or domain shift) phenomenon [7, 8, 9], which results in a degradation of the performance of the method.

A typical countermeasure to this phenomenon is the fine tuning of the method using annotated data from different acoustic conditions. For example, one can retrain a method given a newly collected dataset. But, the annotation of audio data is a tedious process and it is more likely for one to have audio data but not having their annotations. To leverage knowledge from new and unlabeled data, one can use domain adaptation processes. Domain adaptation is a subspace alignment problem, where the goal is the alignment of the latent representations of the data coming from different domains [8, 10, 11, 12]. The impact of the domain adaptation is greater when none or few annotations (labels) exist for data from the different domains. These processes are referred as unsupervised and semi-supervised, respectively, domain adaptation.

Before the emergence of adversarial training, different approaches had been employed to cope with the problem of covariate shift across domains, e.g. kernel mean matching (KMM) [13, 14] and autoencoder scheme based approaches [15, 16, 17]. One of the first works in adversarial domain adaptation with deep neural networks is [18] where the classification and the alignment of the latent representations can occur at the same time. The alignment is performed by using the reverse gradient of the domain classification to optimize the parameters that produce the latent representation for the classification. A similar concept has been adopted in many subsequent works, e.g. [19]. Later, an adversarial domain adaptation approach is presented in [9], where the training procedure of classification and adaptation are not happening simultaneously. The first step obtains a non-adapted model and, in a second step, this model is adapted. This increases the performance of adaptation, compared to the previous existing methods. Another method used in [20] implements classification and reconstruction by employing three different feature extractors and one shared encoder. One of the feature extractors is shared between the domains, while the other two are domain exclusive. The classifier predicts the labels based on the shared features between domains. In addition, there is an adversarial objective function for shared features to help the adaptation by increasing the similarity of extracted features across domains. Another recent work [21] presents two models for source and target while regularizing their parameters by sharing a loss between each layer, targeting to mitigate the existing disparity between source and target distributions. The above methods evaluate the domain adaptation in the context of natural language processing, sentiment classification, and image classification. There are no previous studies in the context of acoustic scene classification.

Driven by the above, in this paper we present the first approach for unsupervised domain adaptation for acoustic scene classification. We investigate the unsupervised domain adaptation scenario, i.e. the acoustic scene labels of the new data are not known during the adaptation part. We use the data from the DCASE 2018 Task 1, subtask B, which consist of recordings from mismatched recording devices [22]. We consider the difference in the acoustic channel, imposed by the different recording devices, as the domains. To mitigate this difference, we introduce a model agnostic process where we encourage the model to match the distributions of the learned representations of the data coming from the annotated (source domain) and the non-annotated (target domain) sets. The contributions of this paper are the following:

1. We follow a recently proposed general framework for adversarial domain adaptation [9] and we alter it by introducing

[*]Equally contributing authors

extra learning signals during the adaptation process;

2. We present the first application of deep neural network based, unsupervised domain adaptation for acoustic scene classification showing the effect of leveraging unlabeled data for acoustic scene classification, through unsupervised domain adaptation.

The rest of the paper is organized as follows. In Section 2 we explain the proposed method, and in Section 3 we present the evaluation procedure that we followed, including the presentation of the dataset used and the models implemented, and the details of the training and testing procedures. The obtained results are reported and discussed in Section 4, followed by the conclusions and proposals for future work in Section 5.

2. PROPOSED DOMAIN ADAPTATION METHOD

The data from the source domain are classified according to a specific set of labeled acoustic scenes. For example, in our study, these acoustic scenes are *airport*, *bus*, *airport*, *metro*, *metro station*, *park*, *public square*, *shopping mall*, *street pedestrian*, *street traffic*, and *tram*. The goal is to assign the target domain data to this same set of labels. The source and target domains data are time-frequency representation of audio (e.g. log mel-band energies). We follow the general framework for adversarial domain adaptation in [9] and choose not to tie the parameters of the source and adapted (target) models. Our models are neural networks that are used to extract the discriminative latent representation of the input data. This representation is used for the label classification by the classifier. The source model is the model optimized with the source data and the target model is the one adapted to the target data. Our presented method is independent of the architecture of the utilized model and concerns the adaptation of a model optimized on the source domain, to the target domain. For this reason, in this section, we present the method for the domain adaptation, and in Section 3 we present the specific models employed.

Having annotated (i.e. with reference labels) data from the source domain, $\mathbf{X}^S = \{\mathbf{X}_1^S, \mathbf{X}_2^S, \ldots, \mathbf{X}_{N_S}^S\}$, and the non-annotated target domain data, $\mathbf{X}^T = \{\mathbf{X}_1^T, \mathbf{X}_2^T, \ldots, \mathbf{X}_{N_T}^T\}$, the goal is to regularize a model M to produce feature mappings of the source domain, $M(\mathbf{X}^S)$, and of the target domain, $M(\mathbf{X}^T)$, that exhibit the same distribution. Then a classifier, trained on $M(\mathbf{X}^S)$, can be used in order to classify $M(\mathbf{X}^T)$. For this process, we employ three steps. At the first step, we pre-train the model M and the classifier C using dataset $\mathbf{X}^S$. Then, at the second step, we use adversarial training (as in generative adversarial network (GAN) [23]) to match the distributions of $M(\mathbf{X}^S)$ and $M(\mathbf{X}^T)$. Finally, at the third step, we test the performance of the classifier on the $M(\mathbf{X}^T)$. All the three steps of the process are schematically illustrated at Figure 1.

We differ from the original proposal of the general framework for adversarial domain adaptation in [9] by utilizing the label classifier C also during the adaptation step. Also, we differ from proposals with gradient reversing, e.g. [8], because the classifier C is not the domain classifier but the label one. We experimentally found that, for our task, the original setup (i.e. without C in the adaptation step) cannot work. In this setup, the adapted model was exhibiting worse label classification performance to both the target and the source domains, compared to the non-adapted one. Observing the adaptation process, we hypothesized that the learning signals used in [9] were not able to drive the model M to produce feature mappings that can be used for later target domain classification from C. Thus, we utilize the C in order to provide an additional learning signal during the adaptation process. This results in more stable domain adaptation process and the adapted model exhibits increased performance at the target domain, compared to the non-adapted one.

We start by having the data from the source domain, $\mathbf{X}^S$, and their corresponding one-hot-encoded labels for the acoustic scene, $\mathbf{Y}^S = \{\mathbf{y}_1^S, \mathbf{y}_2^S, \ldots, \mathbf{y}_{N_S}^S\}$. The first goal is to obtain a model and a classifier that are able to classify the source data (i.e. label and not domain classification). To this end, we utilize the $\mathbf{X}^S$ and $\mathbf{y}^S$ to train our source domain model, M_S, and pretrain the classifier C, by minimizing the loss

$$\mathcal{L}_S = -\sum_{n=1}^{N_S} \mathbf{y}_n^S \log(C(M_S(\mathbf{X}_n^S))), \tag{1}$$

At the second step, we target to obtain a model that can produce mappings of the data from the source and target domains, that they are as close as possible in terms of their distribution. Since we do not have the labels of the target domain, we can only leverage knowledge from the source data and their labels, and from the data of the target domain. Adopting the approach in [9], we use the adversarial training to match the distributions of $M(\mathbf{X}^S)$ and $M(\mathbf{X}^T)$. Specifically, we use an additional, target domain model, M_T, having the same architecture and amount of parameters as M_S. We do not use any constraints between M_S and M_T (e.g. parameters sharing/coupling between M_S and M_T), but we initialize the parameters of M_T with the ones from M_S. Additionally, we use a domain discriminator D that will be optimized to identify if its input is coming from the distribution of the source or the target domain (hence, its output is an indication if its input was or not from the source domain).

We jointly optimize the M_T and D in order to enforce the distribution of the $M_T(\mathbf{X}^T)$ to be as close as possible to the distribution of the $M_S(\mathbf{X}^S)$. In the GAN terminology, one can think the M_T as the generator, the M_S as the real examples, and the D as the discriminator. The output of the generator and real examples are given as an input to the discriminator, and the latter is optimized to identify which is real and which is coming from the generator. At the same time, the generator is optimized to fool the discriminator in believing that the output of the generator is also a real example. In our method, $M_S(\mathbf{X}^S)$ (real examples) and $M_T(\mathbf{X}^T)$ (generator output) are given as an input to the discriminator D. The latter is optimized to identify if its input is $M_S(\mathbf{X}^S)$ or $M_T(\mathbf{X}^T)$. At the same time, we optimize M_T in order to fool D that $M_T(\mathbf{X}^T)$ is $M_S(\mathbf{X}_S)$. We minimize the losses

$$\mathcal{L}_D = -\sum_{n=1}^{N_S} (\log D(M_S(\mathbf{X}_n^S) + \log(1 - D(M_T(\mathbf{X}_n^T))) \text{ and} \tag{2}$$

$$\mathcal{L}_{M_T} = -\sum_{n=1}^{N_S} (\log D(M_T(\mathbf{X}_n^T) + \mathbf{y}_{S_n} \log(C(M_T(\mathbf{X}_n^S))). \tag{3}$$

$\mathcal{L}_D$ is minimized w.r.t D and the $\mathcal{L}_{M_T}$ w.r.t. M_T. In the case where $N_T < N_S$ or $N_T > N_S$, then $\mathbf{X}^T$ will be either oversampled or undersampled, respectively. The minimization of $\mathcal{L}_D$ and $\mathcal{L}_{M_T}$ can be performed jointly or in an alternating way, e.g. do an update of D towards minimizing $\mathcal{L}_D$, then update M_T towards minimizing $\mathcal{L}_{M_T}$, and repeat until some criterion is met. The total loss, e.g. $\mathcal{L}_D + \mathcal{L}_{M_T}$, is a typical minimax objective for adversarial training as it has been used in [9, 12].The actual implementation of the minimization process is tied to the employed models of M_S, M_T,

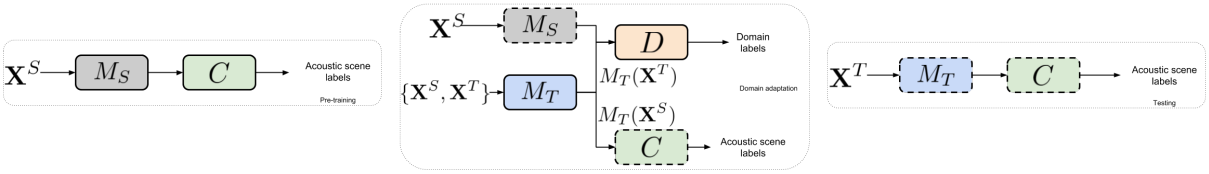


Figure 1: Illustration of the three steps of the domain adaptation method: pre-training; adversarial domain adaptation; and testing. Solid lines indicate the models that are optimized in the corresponding steps, and dashed lines indicate the models that are not optimized.

D, and C, and the dynamics of the training process. Our followed procedure is presented in Section 3. Finally, we use M_T with C in order to classify the $\mathbf{X}^T$.

3. EVALUATION

To assess the performance of our method we focus on the task of acoustic scene classification. We employ a freely available dataset that provides audio data recorded with mismatched recording devices. For model M, we employ two different models; one that achieved the first place in an acoustic scene classification contest[1] and a second that is the published baseline model for the Task 1, subtask B, of the DCASE challenge 2018 [22]. For the rest of the paper, we will refer to the former model as the Kaggle model and to the latter as the DCASE model. All hyper-parameters reported in this section are the same as in the proposed models. All models are implemented using the freely available PyTorch framework[2] and our code can be found online.[3]

3.1. Dataset and data preprocessing

The dataset used for the development and evaluation of our method is the one provided as the development dataset of Task 1, subtask B, of the DCASE 2018 challenge [22]. The dataset is collected with three different recording devices. The main recording device which is referred to as device A consists of a binaural microphone and a recorder using 48 kHz sampling rate and 24 bit resolution. The data from this device were re-sampled and averaged into a single channel to match the characteristic of data recorded by device B and C. The rest of data have been recorded using customer devices such as smart phones and cameras which are referred to as device B and C. This dataset contains a total of 28 hours of audio out of which 24 hours are from device A, 2 hours from device B, and 2 hours from device C. The proposed evaluation setup by the organizers of the DCASE 2018 challenge, 30% of audio files of device A, and 25% of device B and C are dedicated to the validation set.

During the development of our method, the annotations of evaluation/test data of the Task 1, subtask B, were not publicly available. Therefore, we use the original (i.e. proposed by the evaluation setup of the DCASE Task 1, subtask B) validation data as our test data (referred to as test data for the rest of this paper), a randomly selected 10% of the original training data as our validation (referred to as validation data from now on), and the rest of the original training data as our actually training data (referred to as training data from now on). This means that we use 5510 files from device A, 486 files from device B, and 486 files from device C as training data. 612, 54, and 54 files from device A, B, and C, respectively, are used as our validation set. We test our method on 2518, 180, and 180 files, from devices A, B, and C respectively which is equivalent to original validation set of the Task 1, subtask B.

From the available files, we extracted 64 log Mel-band energies, using a 2048 samples ($\sim$ 46 ms) Hamming window and 50% overlap. The extracted features from the data recorded from device A is our source domain data. The rest (B and C) are our target domain data. We use the Librosa package for feature extraction.[4] Since the amount of data for the target domain (i.e. the data from the B and C devices) are less than the source domain data (i.e. the data from device A), we oversampled the data from target domain to have the same amount of training data as the number of samples from device A, approximately 5.6 times more than the original size.

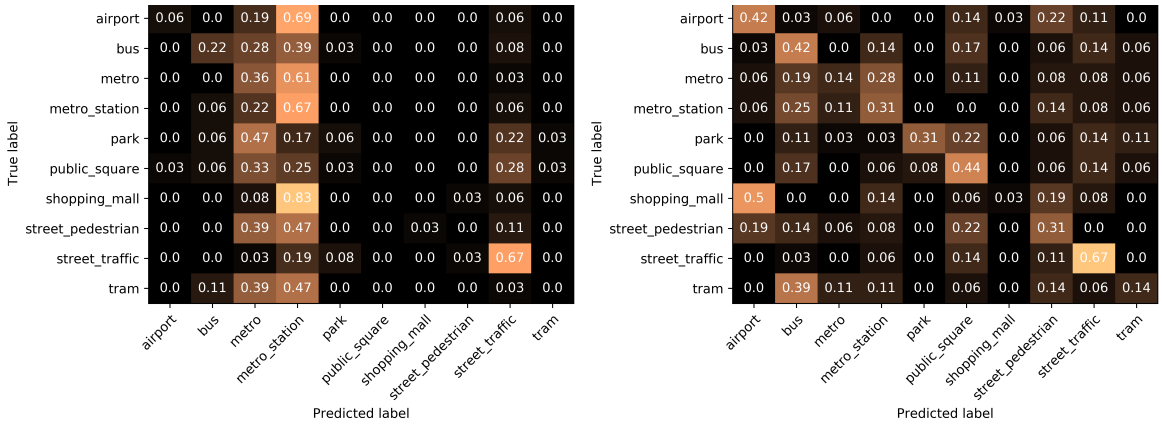
3.2. Models used

Since our proposed method is independent of the employed model, we evaluate it on two different and published models. The first is the Kaggle model and the second is DCASE baseline model. The Kaggle model is mainly a convolutional neural network (CNN) which has 5 convolutional layers, with kernel sizes of {(11, 11), (5, 5), (3, 3), (3, 3), (3,3)} and amount of channels/filters of {48, 128, 192, 192, 128}. The first two convolutional layers use strides of (2,3) and the rest (1,1). The first two convolutional layers together with the last one are followed by rectified linear unit (ReLU) non-linearity, max pooling layer, and batch normalization. The rest convolutional layers are followed only by the ReLU non-linearity. DCASE model consists of two convolutional layers with 32 and 64 filters, respectively. Both layers have a (7, 7) kernel size followed by batch normalization, ReLU non-linearity, and a max pooling operation. The kernels of the pooling operations are {(5, 5), (4, 100)}.

We use 64 log mel-band energies, but for the development of the DCASE model, 40 mel band energies were used. Therefore, we had to slightly alter the DCASE model in order to utilize our data. Specifically, we altered the kernel of the first pooling operation and the padding of the second convolutional layer. That is, we used kernel size of (8,4) for the pooling operation and we specified the padding for the second convolutional layer at (3,0). Because the Kaggle and the DCASE models had a different dimensionality of their outputs, we used two different discriminators.

As the discriminator D, when employing the Kaggle model, we use three convolutional layers all with a kernel size of (3,3) and {64, 32, 16} as the number of channels/filters. All layers are followed by the ReLU non-linearity and batch normalization. The output of the third convolutional layer is flattened and given as an input to a linear layer, which outputs the prediction for samples as source or target. As a discriminator for the DCASE model we used one linear layer. The input to our label classifier C is the output of the last layer of the model M which is turned to a vector (i.e. flattened) and is given as an input to three for the Kaggle and two for the DCASE model linear layers followed by 25%, for the Kaggle model, and 30%, for

[1] https://www.kaggle.com/c/acoustic-scene-2018
[2] https://pytorch.org/
[3] https://github.com/shayangharib/AUDASC
[4] https://librosa.github.io/librosa/



(a) Confusion matrix for the non-adapted Kaggle model (b) Confusion matrix for the adapted Kaggle model

Figure 2: Confusion matrix of the **non-adapted**, (a), and **adapted**, (b), **Kaggle model** for the **target domain**. The values are normalized according to the amount of examples in each class. Brighter color indicates higher value.

the DCASE model, dropout. The non-linearity of all except the last layer of the classifier is the ReLU. Lastly, the output of our classifier is followed by a softmax non-linearity.

3.3. Training and testing procedure

For the pre-training step, we use a minibatch size of 38 samples, all selected from source domain. During the domain adaptation process we used a minibatch size of 16 samples, out of which 10 were selected from the source domain and 6 from the target domain (more specifically 3 from device B and 3 from device C). For the pre-training and the domain adaptation process, Adam was selected as the optimizer with learning rate of $1e{-}4$ and other values according to the ones presented in the original paper [24]. We updated the parameters of the M_S and M_T after each iteration but (according to experimental observations) we updated the parameters of the discriminator D after 10 iterations. We stopped the optimization procedure in pre-training and domain adaptation processes after 350 and 300 epochs respectively.

4. RESULTS AND DISCUSSION

We report the obtained accuracy of the label classification when using the non-adapted models (i.e. in the pre-training step) and when using the models after the domain adaptation process (i.e. adapted models), using the data from source and target domains. Table 1 presents the obtained accuracy on the source and target domains when using the Kaggle model and when using the DCASE model, respectively. Additionally, we present the confusion matrices of the label classification for the target domain of the non-adapted and adapted Kaggle model in Figure 2, where can be seen that the adapted model manages to increase significantly the correctly classified examples from all labels. This is easily visualized by the diagonal of the confusion matrices, where in Figures 2a and 2b there is a considerable difference. Additionally, our proposed method manages to increase the performance of the classification for different models. That is, no matter the architecture of the model M, by following our proposed method there is an increase on the target domain without significant decrease in the performance for the source domain. In fact, we managed to increase also the performance on the source domain for the adapted model. This is apparent in Table 1, where the obtained accuracy for the adapted models and the target domain is greater, compared to the non-adapted. Furthermore, from the same table can be seen that the reduction in the accuracy at the source domain is around 0.5% for the DCASE model. The accuracy is marginally (i.e. $\sim 0.1\%$) greater for the source domain and the adapted Kaggle model (i.e. 65.25% to 65.37%). We attribute this small increase to the usage of the label classifier C during the domain adaptation process.

Table 1: Obtained accuracy for the non-adapted and adapted Kaggle and DCASE models.

	Kaggle model		DCASE model	
	Non adapted	**Adapted**	**Non adapted**	**Adapted**
Source	65.25%	65.37%	61.71%	61.23%
Target	20.28%	31.67%	19.17%	25.28%

5. CONCLUSIONS AND FUTURE WORK

We presented the first unsupervised adversarial domain adaptation for acoustic scene classification, which is also independent of the actual models used. The goal of our method is the adaptation of a pre-trained model on a source dataset, to a new and unseen target dataset. In a GAN-like setting, the adapting model tries to fool a discriminator that its output comes from the source dataset, while the non-adapted model informs the discriminator about the data that really coming from the source dataset.

We managed to increase the performance of the used models to the unseen dataset by approx. 10%. This indicates that the domain adaption approaches can provide an appealing solution for the problem of mismatched training and testing data, regarding the acoustic scene classification. As future directions we suggest the adoption of different GAN losses and the usage of domain adaptation for sound event detection.

6. ACKNOWLEDGMENT

S. Gharib, E. Çakir, K. Drossos, and T. Virtanen wish to acknowledge the CSC-IT Center for Science, Finland, for computational resources. Part of the computations leading to these results was performed on a TITAN-X GPU donated by NVIDIA to K. Drossos. Part of the research leading to these results has received funding from the European Research Council under the European Unions H2020 Framework Programme through ERC Grant Agreement 637422 EVERYSOUND.

7. REFERENCES

[1] M. Valenti, A. Diment, G. Parascandolo, S. Squartini, and T. Virtanen, "Dcase 2016 acoustic scene classification using convolutional neural networks," *IEEE AASP Challenge on Detection and Classification of Acoustic Scenes and Events (DCASE2016), Budapest, Hungary*, 2016.

[2] Y. Han, J. Park, and K. Lee, "Convolutional neural networks with binaural representations and background subtraction for acoustic scene classification," *the Detection and Classification of Acoustic Scenes and Events (DCASE)*, pp. 1–5, 2017.

[3] Y. Han and J. Park, "Convolutional neural networks with binaural representations and background subtraction for acoustic scene classification," DCASE2017 Challenge, Tech. Rep., September 2017.

[4] Z. Weiping, Y. Jiantao, X. Xiaotao, L. Xiangtao, and P. Shaohu, "Acoustic scene classification using deep convolutional neural network and multiple spectrograms fusion," DCASE2017 Challenge, Tech. Rep., September 2017.

[5] B. Lehner, H. Eghbal-Zadeh, M. Dorfer, F. Korzeniowski, K. Koutini, and G. Widmer, "Classifying short acoustic scenes with I-vectors and CNNs: Challenges and optimisations for the 2017 DCASE ASC task," DCASE2017 Challenge, Tech. Rep., September 2017.

[6] S. Park, S. Mun, Y. Lee, and H. Ko, "Acoustic scene classification based on convolutional neural network using double image features," DCASE2017 Challenge, Tech. Rep., September 2017.

[7] A. Gretton, A. Smola, J. Huang, M. Schmittfull, K. Borgwardt, and B. Schölkopf, *Covariate shift and local learning by distribution matching*. Cambridge, MA, USA: MIT Press, 2009, pp. 131–160.

[8] F. Alam, S. Joty, and M. Imran, "Domain adaptation with adversarial training and graph embeddings," in *32nd International Conference on Machine Learning (ICML 2015)*, vol. 37, Jul. 2015, pp. 1180–1189.

[9] E. Tzeng, J. Hoffman, K. Saenko, and T. Darrell, "Adversarial discriminative domain adaptation," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul. 2017, pp. 2962–2971.

[10] Z. Pei, Z. Cao, M. Long, and J. Wang, "Multi-adversarial domain adaptation," in *32nd AAAI Conference on Artificial Intelligence*, Feb. 2018.

[11] B. Fernando, A. Habrard, M. Sebban, and T. Tuytelaars, "Unsupervised visual domain adaptation using subspace alignment," in *2013 IEEE International Conference on Computer Vision*, Dec. 2013, pp. 2960–2967.

[12] Y. Ganin and V. Lempitsky, "Unsupervised domain adaptation by backpropagation," in *56th Annual Meeting of the Association for Computational Linguistics (ACL)*, Jul. 2015.

[13] A. Gretton, A. J. Smola, J. Huang, M. Schmittfull, K. M. Borgwardt, and B. Schölkopf, "Covariate shift by kernel mean matching," 2009.

[14] Y.-Q. Miao, R. Araujo, and M. S. Kamel, "Cross-domain facial expression recognition using supervised kernel mean matching," in *Machine Learning and Applications (ICMLA), 2012 11th International Conference on*, vol. 2. IEEE, 2012, pp. 326–332.

[15] J. Deng, Z. Zhang, F. Eyben, and B. Schuller, "Autoencoder-based unsupervised domain adaptation for speech emotion recognition," *IEEE Signal Processing Letters*, vol. 21, no. 9, pp. 1068–1072, 2014.

[16] X. Glorot, A. Bordes, and Y. Bengio, "Domain adaptation for large-scale sentiment classification: A deep learning approach," in *Proceedings of the 28th international conference on machine learning (ICML-11)*, 2011, pp. 513–520.

[17] M. Chen, Z. Xu, K. Weinberger, and F. Sha, "Marginalized denoising autoencoders for domain adaptation," *arXiv preprint arXiv:1206.4683*, 2012.

[18] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky, "Domain-adversarial training of neural networks," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 2096–2030, 2016.

[19] Z. Pei, Z. Cao, M. Long, and J. Wang, "Multi-adversarial domain adaptation," in *AAAI Conference on Artificial Intelligence*, 2018.

[20] K. Bousmalis, G. Trigeorgis, N. Silberman, D. Krishnan, and D. Erhan, "Domain separation networks," in *Advances in Neural Information Processing Systems*, 2016, pp. 343–351.

[21] A. Rozantsev, M. Salzmann, and P. Fua, "Beyond sharing weights for deep domain adaptation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2018.

[22] A. Mesaros, T. Heittola, and T. Virtanen, "A multi-device dataset for urban acoustic scene classification," *ArXiv e-prints*, July 2018.

[23] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2014, pp. 2672–2680. [Online]. Available: http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf

[24] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014.

ACOUSTIC BIRD DETECTION WITH DEEP CONVOLUTIONAL NEURAL NETWORKS

Mario Lasseck

Museum fuer Naturkunde
Leibniz Institute for Evolution and Biodiversity Science
Invalidenstraße 43, 10115 Berlin, Germany
Mario.Lasseck@mfn.berlin

ABSTRACT

This paper presents deep learning techniques for acoustic bird detection. Deep Convolutional Neural Networks (DCNNs), originally designed for image classification, are adapted and fine-tuned to detect the presence of birds in audio recordings. Various data augmentation techniques are applied to increase model performance and improve generalization to unknown recording conditions and new habitats. The proposed approach is evaluated on the dataset of the Bird Audio Detection task which is part of the IEEE AASP Challenge on Detection and Classification of Acoustic Scenes and Events (DCASE) 2018. It surpasses previous state-of-the-art achieving an area under the curve (AUC) above 95 % on the public challenge leaderboard.

Index Terms— Bird Detection, Deep Learning, Deep Convolutional Neural Networks, Data Augmentation

1. INTRODUCTION

Automated bird detection is an important tool for acoustic wildlife monitoring. It can serve as a first step to filter large datasets and reduce human as well as computational effort by focusing on regions of interest with bird activity before conducting further analysis like e.g. species identification or population estimation.

In the DCASE 2018 Bird Audio Detection challenge participants are asked to decide whether or not there are any birds present in short excerpts of audio recordings. For training, three development datasets are provided recorded in different parts of the world. The test data is recorded in monitoring scenarios not matching the training data making it necessary to develop models inherently generalizing well to unknown recording conditions and new habitats. The task is an expanded version of the Bird Audio Detection challenge which ran in 2016/2017. An overview and further details about task and data provided for training and evaluation are given in [1] and [2].

2. IMPLEMENTATION DETAILS

To detect the presence/absence of bird sounds in audio recordings the best model of the LifeCLEF 2018 [3] Bird Identification Task [4] is adapted to binary classification. The original model designed to identify 1500 bird species is described in [5].

2.1. Data Preparation

All audio files of the development and evaluation sets are first pre-processed by applying a shallow high pass filter (Q = 0.707) with a cutoff frequency at 2 kHz and furthermore resampled to 22050 Hz. The filter reduces low frequency energy improving signal-to-noise ratio for frequency bands relevant to bird sounds. Downsampling reduces the amount of data to process without losing too much relevant acoustic information.

2.2. Training Setup

To develop an acoustic bird detection system, DCNNs pre-trained on ImageNet [6] are fine-tuned with mel spectrogram images representing short audio chunks. Training is done via PyTorch [7] utilizing PySoundFile and librosa [8] python packages for audio file reading and processing. The same basic pipeline as for the BirdCLEF 2018 task is used for data loading and can be summarized as follows:

- extract audio chunk from file with duration of ca. 4 seconds
- apply short-time Fourier transform
- convert to mel spectrogram
- remove low and high frequencies
- normalize and convert power spectrogram to decibel units
- resize spectrogram to fit input dimension of the network
- convert grayscale image to RGB image

As recommended by the challenge organizers, two development sets are used for training and the remaining one for performance validation. Since BirdVox-DCASE-20k is larger than the other two datasets combined, it is always part of the training set and therefore only two folds (see Table 1) out of three possible combinations are used for cross-validation:

Table 1: Training/validation splits used for training

Fold	Training sets	Validation set
1	ff1010bird, BirdVox-DCASE-20k	warblrb10k
2	warblrb10k, BirdVox-DCASE-20k	ff1010bird

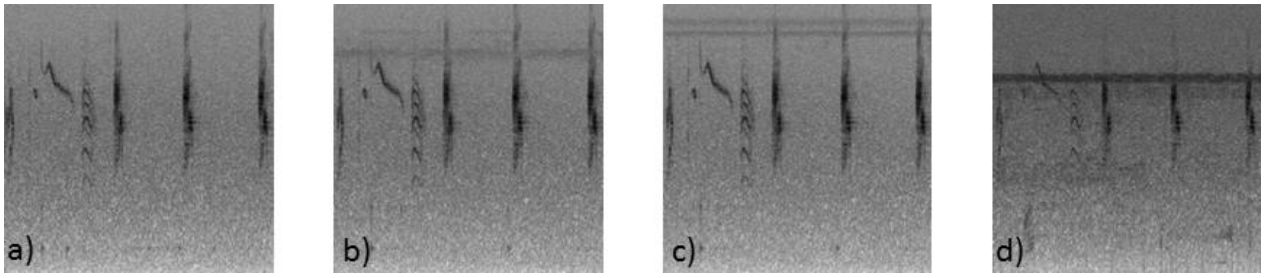


Figure 1: Examples of data augmentation via adding chunks from random files with same label or label 0 (no bird). a) mel spectrogram of original audio chunk without augmentation; b), c) & d) mel spectrogram with augmentation

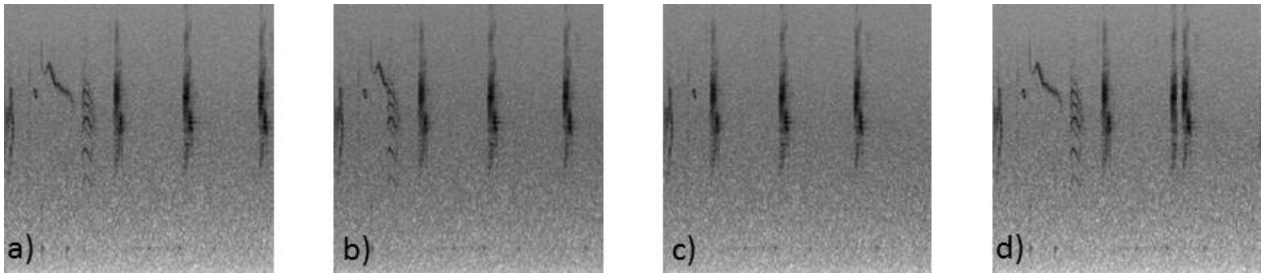


Figure 2: Examples of data augmentation via time interval dropout. a) mel spectrogram of original audio chunk without augmentation; b), c) & d) mel spectrogram with augmentation

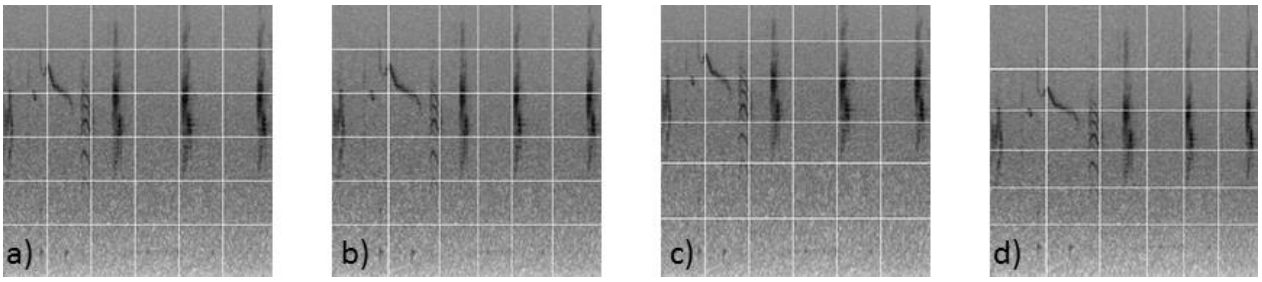


Figure 3: Examples of data augmentation via piecewise time and frequency stretching (with grid overlay for better visualization). a) mel spectrogram of original audio chunk without augmentation; b) local time stretching; c) local frequency stretching; d) combined local time and frequency stretching

In each training epoch all files of the training set are processed in random order to extract audio chunks at random position. Training is done with a batch size between 80 and 90 samples using up to three GPUs (Nvidia Geforce 1080 and 1080 Ti). Categorical cross entropy is utilized as loss function and stochastic gradient descent as optimizer with Nesterov momentum 0.9, weight decay 1e-4 and a constant learning rate of 0.01. For validation and test sets audio chunks are extracted successively from each file with an overlap of 10 % for validation files during training and 50 % for files in the evaluation test set. Predictions are summarized for each file by taking either the mean or maximum over all chunk predictions per file.

2.3. Data Augmentation

To increase model performance and improve generalization to new recording conditions and habitats, various data augmentation techniques are applied in both time and frequency domain. The following methods are applied in time domain regarding audio chunks:

- apply jitter to chunk duration (ca. ± 0.4 s)
- extract chunks from random position in file (wrap around if end of file is reached and continue from beginning)
- add 3 audio chunks from random files with label 0 (no bird)
- add 2 audio chunks from random files with the same label
- apply random factor to signal amplitudes of all chunks before summation (superposition)
- apply random cyclic shift
- apply time interval dropout by skipping random number of samples

The audio chunk (or sum of chunks) is then transformed to frequency domain via short-time Fourier transform with a window size of 1536 samples and a hop length of 360 samples.

Table 2: Model properties

Model ID	M1	M2	M3	M4	M5	M6
Included in submission	1	2,3,4	3,4	3,4	3,4	4
Network architecture	Inception	Inception	Inception	Inception	Inception	ResNet
Chunk duration [s]	4.0	4.0	4.0	3.5	4.0	4.0
Chunk duration jitter [s]	0.45	0.45	0.45	0.4	0.4	0.45
Chance to add 1st chunk w. label 0 [%]	75	75	75	80	70	75
Chance to add 2nd chunk w. label 0 [%]	75	75	50	80	70	75
Chance to add 3rd chunk w. label 0 [%]	50	50	0	60	40	50
Chance to add 1st chunk w. same label [%]	50	50	50	60	60	50
Chance to add 2nd chunk w. same label [%]	50	50	50	50	60	50
Batch size	90	90	90	84	84	78
Training epochs	76	52	177	44	68	217
Pooling method	mean	max	max	max	max	max
Fold	1	2	2	2	2	2
AUC val. set [%]	93.46	93.21	92.22	93.03	92.98	92.01
AUC test subset [%]	90.40	93.66	-	-	-	-

Frequencies are mel scaled with low and high frequencies removed resulting in a spectrogram of 310 mel bands representing a range of approximately 160 to 10300 Hz. Normalization and logarithm is applied to the power spectrogram yielding a dynamic range of approximately 100 dB. The final spectrogram image is resized to 299x299 pixel to fit the input dimension of the InceptionV3 network (or 224x224 pixel for ResNet). Since networks are pre-trained on RGB images, the grayscale image is duplicated to all three color channels. Further augmentation is applied in frequency domain to the spectrogram image during training:

- frequency shifting/stretching by cutting a random number of the first 10 and last 6 rows of the image
- piecewise time/frequency stretching by resizing a random number of columns/rows at random position
- use of different interpolation filters for resizing
- apply color jitter (brightness, contrast, saturation, hue)

Some of the above augmentation techniques were already applied successfully by other teams in previous bird identification or detection tasks like e.g. adding background noise or sounds from files belonging to the same class with random intensity [9,10,11] or applying cyclic shift to the sample array by a random amount [9,11,12]. Pitch or frequency shifting and stretching was previously used in similar ways by e.g. [13], [14] and applying color jitter is very common for training image classifiers. A few augmentation methods, further improving model accuracy and generalization, were newly introduced this year for the LifeCLEF 2018 Bird Identification Task. They also significantly help to increase performance regarding bird detection and are described briefly in the following sections.

With a chance of 30 % **time interval dropout** is realized by skipping a random number of samples (between zero and length of an entire chunk) at a randomly chosen position when reading from the audio file (see Figure 2).

Besides manipulating the duration or speed of an entire chunk, **piecewise time stretching** is applied with a 50 % chance to change speed at multiple times within a chunk. This is accomplished by dividing the spectrogram image in several vertical pieces, each having a width randomly chosen between 10 and 100 pixel. Afterwards, pieces are resized individually by a factor randomly chosen between 0.9 and 1.1 along the horizontal (time) axis. To realize local or **piecewise frequency stretching** the same procedure is applied in an analogous manner to the vertical frequency axis with a 40 % chance and a stretch factor between 0.95 and 1.15 (see Figure 3).

For resizing, the high-quality Lanczos filter of the Python Imaging Library is used by default. However, in 15 % of the cases a **random choice of interpolation filter** is realized using different resampling filters from the library (Nearest, Box, Bilinear, Hamming and Bicubic).

3. RESULTS

Detection performance is evaluated via the area under the curve metric. Scores on validation data and public leaderboard test data are listed in Table 3 along with models or ensemble of models belonging to each DCASE 2018 submission. Properties of individual models are summarized in Table 2. Models mainly differ in duration of chunks, duration jitter and conditional probabilities of chunks superimposed for augmentation. Most models use an InceptionV3 architecture [15] except one which uses a 152-layer residual network (ResNet-152) [16]. For the first two submissions a single model was trained with equal properties but on different training/validation splits (see Table 1): M1 on fold1 and M2 on fold2. M1 gives better performance on the warblrb10k validation set compared to M2 regarding the ff1010bird validation set but M2 performs much better on the public test subset. Also M2 was submitted twice to the public leaderboard using different pooling methods to summarize chunk predictions per audio file. Taking the mean over all chunks results in 93.32 % AUC compared to taking the maximum which obtains an AUC of 93.66 %. As a consequence all following models were trained on fold2 with max pooling of chunk predictions. For the third and fourth DCASE submission different models (see Table 2 and 3) were ensembled by averaging their file-based predictions. For submission five, model M2 was trained from scratch without using pre-trained weights (see discussion section).

Table 3: Submission models and evaluation scores

Submission index	1	2	3	4	5
Submission date	24/7	26/7	28/7	30/7	3/8
Models in ensemble	M1	M2	M2, M3, M4, M5	M2, M3, M4, M5, M6	M2* (not pre-trained)
AUC val. set [%]	93.5	93.2	93.8	93.8	91.0
AUC test set [%]	90.4	93.7	95.0	95.3	91.4

4. DISCUSSION

The BirdCLEF 2018 model designed to identify individual bird species was successfully adapted to the binary classification task detecting bird activity of any kind in audio recordings. Even with a single model, detection performance of more than 93 % AUC can be achieved on unseen data not matching the conditions of the training set. By ensembling models with different properties results can be further improved to above 95 % AUC.

In the following section a few differences between BirdCLEF and DCASE system are pointed out. For bird detection, smaller chunks tend to work better (4s vs. 5s). Interestingly, decreasing the learning rate after a few epochs or squaring chunk predictions before pooling didn't help to further improve detection performance. Also, some augmentation techniques applied for species identification were not used for the bird detection task for example reconstructing the audio signal by mixing individual sound elements or choosing files from different versions of the development set (with/without high pass filtering, artificially degrading audio quality by encoding files to mp3 with low bit rate, removing silent parts containing only background noise, etc.). Without these augmentation methods data preparation is greatly simplified, especially since no segmentation of audio files into signal and noise parts is required. It would be interesting to investigate whether or not these additional techniques are able to further increase detection results. An overview to what extent individual augmentation techniques are able to increase performance on species identification is given in Table 1 in [5]. The most effective augmentation methods are:

1. adding noise/content from random files
2. piecewise time and frequency stretching
3. time interval dropout

These techniques also proved to be successful for the bird detection task (see examples in Figure 1-3).

Other approaches weren't able to further improve detection performance. For example keeping the original sample rate of 44.1 kHz didn't achieve better results but made the training significantly slower. Also, fine-tuning a model pre-trained on the BirdCLEF dataset didn't lead to better results but convergence was much faster during training (92 % AUC in 5 epochs compared to ca. 10 epochs for networks pre-trained on ImageNet).

Although getting rather low scores on the validation set, adding a ResNet-152 based model to the ensemble of submission 3 helped to increase performance of submission 4. The different network architecture seems to complement the Inception predictions quite well.

As mentioned before, models were fine-tuned using neural networks pre-trained on the "trimmed" Large Scale Visual Recognition Challenge (ILSVRC) [17] version of ImageNet, a dataset with almost 1.5 million photographs of 1000 object categories scraped from the web. It is not related to spectrogram images of bird sounds or audio in general. Since it is an external dataset not on the list of pre-registered datasets allowed in the challenge, results of submission 1-4 cannot be directly compared with those of other teams and are not part of the official final challenge scores. However, in a post-challenge experiment model M2 was trained from scratch without transfer learning. It obtains 91.4 % AUC on the public leaderboard (see submission 5 in Table 3 and Figure 4) and provides the best system for the task ranking first place in the official challenge results [18]. This demonstrates, with the presented approach competitive results are possible also without using pre-trained weights. But when adapting off-the-shelf ConvNets for sound detection, starting with pre-trained weights can have some advantages. Training can be significantly faster and even lead to better detection performance, especially in cases where there is only a limited amount of audio data available. Nevertheless, if fine-tuning a network originally designed for image classification, re-training the entire network, not just the last layers is essential. When training the final classification layer of model M2 exclusively, using features from the penultimate layer, only 77 % AUC was obtained on the validation set.

Finally the results of this work show, sound detection benefits from fine-tuning DCNNs pre-trained on large amounts of image data, even if this data comes from a completely different domain.

5. ACKNOWLEDGMENT

I would like to thank Dan Stowell, Hervé Glotin, Yannis Stylianou and Mike Wood for organizing this challenge. I especially want to thank Elias Sprengel for the fruitful cooperation during the previous Bird Audio Detection challenge [19]. I also want to thank the Museum fuer Naturkunde Berlin and the Bundesministerium fuer Wirtschaft und Energie for supporting my research.

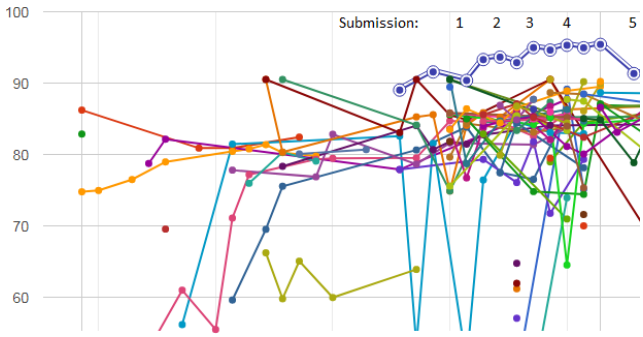


Figure 4: Public leaderboard of the DCASE 2018 Bird Audio Detection challenge. The above described methods and submissions belong to the highlighted and numbered blue dots.

6. REFERENCES

[1] http://dcase.community/challenge2018/

[2] Stowell D, Stylianou Y, Wood M, Pamuła H, Glotin H (2018) Automatic acoustic detection of birds through deep learning: the first Bird Audio Detection challenge. In: Methods in Ecology and Evolution

[3] Joly A, Goëau H, Botella C, Glotin H, Bonnet P, Planqué R, Vellinga WP, Müller H (2018) Overview of LifeCLEF 2018: a large-scale evaluation of species identification and recommendation algorithms in the era of AI, In: Proceedings of CLEF 2018

[4] Goëau H, Glotin H, Planqué R, Vellinga WP, Stefan K, Joly A (2018) Overview of BirdCLEF 2018: monophone vs. soundscape bird identification. In: CLEF working notes 2018

[5] Lasseck M (2018) Audio-based Bird Species Identification with Deep Convolutional Neural Networks. In: Working Notes of CLEF 2018 (Cross Language Evaluation Forum)

[6] Deng J, Dong W, Socher R, Li LJ, Li K, Fei-Fei L (2009) Imagenet: A largescale hierarchical image database. In: IEEE Conference on Computer Vision and Pattern Recognition, 2009. pp. 248–255

[7] Paszke A, Gross S, Chintala S et al. (2017) Automatic differentiation in PyTorch. In: NIPS-W

[8] McFee B, McVicar M, Balke S et al. (2018) librosa/librosa: 0.6.0. Zenodo. http://doi.org/10.5281/zenodo.1174893

[9] Sprengel E, Jaggi M, Kilcher Y, Hofmann T (2016) Audio based bird species identification using deep learning techniques. In: Working notes of CLEF 2016

[10] Kahl S, Wilhelm-Stein T, Hussein H et al. (2017) Large-Scale Bird Sound Classification using Convolutional Neural Networks. In: Working Notes of CLEF 2017

[11] Fazekas B, Schindler A, Lidy T (2017) A Multi-modal Deep Neural Network approach to Bird-song Identification. In: Working Notes of CLEF 2017

[12] Grill T, Schlüter J (2017) Two Convolutional Neural Networks for Bird Detection in Audio Signals. In: 25th European Signal Processing Conference (EUSIPCO2017). Kos, Greece. https://doi.org/10.23919/EUSIPCO.2017.8081512

[13] Sevilla A, Glotin H (2017) Audio bird classification with inception-v4 extended with time and time-frequency attention mechanisms. In: Working Notes of CLEF 2017

[14] Fritzler A, Koitka S, Friedrich CM (2017) Recognizing Bird Species in Audio Files Using Transfer Learning. In: Working Notes of CLEF 2017

[15] Szegedy C, Vanhoucke V, Ioffe S (2015) Rethinking the Inception Architecture for Computer Vision. arXiv preprint arXiv:1512.00567

[16] He K, Zhang X, Ren S, Sun J (2016) Deep Residual Learning for Image Recognition. In: CVPR, 2016

[17] Russakovsky O, Deng J et al. (2015) ImageNet Large Scale Visual Recognition Challenge. IJCV, 2015

[18] http://dcase.community/challenge2018/task-bird-audio-detection-results

[19] Stowell D, Wood M, Stylianou Y, Glotin H (2016) Bird detection in audio: a survey and a challenge. arXiv preprint arXiv:1608.03417

VOCAL IMITATION SET: A DATASET OF VOCALLY IMITATED SOUND EVENTS USING THE AUDIOSET ONTOLOGY

Bongjun Kim[1], *Madhav Ghei*[2], *Bryan Pardo*[3], *Zhiyao Duan*[4]

[1 2 3] Dept. of Electrical Engineering and Computer Science, Northwestern University, Evanston, IL, USA,
[1]bongjun@u.northwestern.edu, [2]madhavghei2018@u.northwestern.edu, [3]pardo@northwestern.edu
[4] Dept. of Electrical and Computer Engineering, University of Rochester, Rochester, NY, USA,
zhiyao.duan@rochester.edu

ABSTRACT

Query-By-Vocal Imitation (QBV) search systems enable searching a collection of audio files using a vocal imitation as a query. This can be useful when sounds do not have commonly agreed-upon text-labels, or many sounds share a label. As deep learning approaches have been successfully applied to QBV systems, datasets to build models have become more important. We present Vocal Imitation Set, a new vocal imitation dataset containing $11,242$ crowd-sourced vocal imitations of 302 sound event classes in the AudioSet sound event ontology. It is the largest publicly-available dataset of vocal imitations as well as the first to adopt the widely-used AudioSet ontology for a vocal imitation dataset. Each imitation recording in Vocal Imitation Set was rated by a human listener on how similar the imitation is to the recording it was an imitation of. Vocal Imitation Set also has an average of 10 different original recordings per sound class. Since each sound class has about 19 listener-vetted imitations and 10 original sound files, the data set is suited for training models to do fine-grained vocal imitation-based search within sound classes. We provide an example of using the dataset to measure how well the existing state-of-the-art in QBV search performs on fine-grained search.

Index Terms— Vocal imitation datasets, audio retrieval, query-by-vocal imitation search

1. INTRODUCTION

Imitating sounds with one's voice is a natural and effective way of delivering an audio concept in human-to-human communication. It can be even more effective than describing sound with words, when it is not clear how to describe the sound using words [1, 2]. This communication is possible because a human listener can identify what the imitation represents. If a machine can understand a human's vocal imitation, users can interact with the machine in this natural way for various audio-related tasks, such as sound designing [3, 4, 5, 6], or searching for melody in a music database by humming the desired melody [7].

Vocal imitations have recently gotten attention as a query method for general sound event search [8, 9, 10]. Commercially-deployed sound search and retrieval systems for general audio (e.g., Soundcloud [1], Freesound [2]) rely on text-based search. Text search fails when there is no search-relevant metadata about the audio content in the file. Text search may also be insufficient when one wants to retrieve a sound that does not have a commonly agreed-upon label or has a label unknown to the user (e.g., a new synthesizer sound). Search using a text label also often produces too many examples (e.g., "dog bark" producing over 1000 examples of dogs barking) and does not provide the specificity required (e.g., the particular bark of a frightened beagle). Using a vocal imitation as the search query, known as Query by Vocal Imitation (QBV), the user can provide information about the desired audio in a way complimentary to text querying.

QBV systems compare the vocal imitation to the content of each audio file in a collection. As deep neural networks have become a typical approach to sound classification tasks [11], they also have been successfully applied to QBV [10, 12]. However, while researchers have put significant effort into developing datasets for various sound classification tasks such as the DCASE dataset [13], the Urban Sound dataset [14], and AudioSet [15], developing datasets for QBV systems has had less attention. This is probably because collecting vocal imitation datasets requires much more human effort. Mehrabi et.al [16] created a dataset of 420 vocal imitations of 30 drum samples, which is useful for a musician to search for drum sounds, but not broad enough in coverage to train general-purpose QBV retrieval systems.

Cartwright and Pardo [17] created the VocalSketch dataset which covers more varieties of sound classes. It includes 240 reference recordings in 4 broad groups: Acoustic Instruments (40), Commercial Synthesizers (40), Everyday Sound (120), and Single Synthesizer (40). Each reference recording has about 10 imitations collected through Amazon Mechanical Turk where the participants were asked to listen to reference recordings (e.g., sound of dog barking) and imitate them vocally. Although the dataset has been successfully used to build QBV retrieval systems [10, 18, 12], it has only $2,400$ vocal imitations made in direct response to an audio file. This is much smaller than other environmental sound datasets and might be insufficient for training a deep model, which often contains many more parameters than the number of vocal imitations in this dataset. VocalSketch dataset also contains only one reference audio file per sound class (e.g. just one "dog barking" file). This means that systems trained on VocalSketch dataset can only learn coarse-grained distinctions between broad classes of sound ("dog barking" vs "violin"), as opposed to fine-grained within-class search (the right dog bark from a set of many dog barks).

In this work, we introduce *Vocal Imitation Set*, a new crowd-sourced vocal imitation dataset. Vocal Imitation Set has more than double the number of imitations available in VocalSketch dataset.

This work was supported by NSF Grants 1617497 and 1617107.

[1] https://soundcloud.com/

[2] https://freesound.org/

Vocal Imitation Set has $11,242$ recordings consisting of $5,601$ high-quality imitations that passed our inclusion criteria, as well as an additional $5,641$ draft, training, and excluded imitations. Vocal Imitation Set is the first dataset of vocal imitations that uses a widely-used ontology. Its sound classes were selected from the AudioSet ontology [15]. Each high-quality imitation was also rated by a human evaluator for perceptual similarity to the audio file it was an imitation of. Perceptual similarity ratings could be very useful in building and testing QBV retrieval systems. Lastly, Vocal Imitation Set has a mean of 10 different original recordings per sound class (e.g., 10 distinct police siren recordings). This will enable the development of fine-grained vocal imitation-based search algorithms, which are more useful when a database has multiple sound events with the same text tags.

2. DATA COLLECTION

2.1. Reference audio collection

Since our goal is to create a vocal imitation dataset that can be used to build a general-purpose QBV search system, the set of sound classes should cover a wide range of sound events. Therefore, we selected sound classes from the AudioSet ontology [15]. This ontology contains 632 sound classes that are structured hierarchically with a maximum depth of 6 levels. The top-level categories include *Animal sounds*, *Channel/environment/background sounds*, *Human sounds*, *Music*, *Natural sounds*, *Sounds of things*, and *Source-ambiguous sounds*. The sound classes in the AudioSet ontology were manually curated to represent a broad set of audio events one might encounter in real-world recordings and each class is assumed to be distinguishable from other classes based on sound alone without any additional information (e.g., visual cue or details of context). For each sound class, AudioSet provides links to YouTube videos that were tagged with the text label for that class. The audio tracks from these videos typically contains multiple, overlapping sounds. Perhaps for this reason, audio from these YouTube videos has been widely used as a benchmark dataset for sound event detection and scene classification [19, 20]. For more details about AudioSet, refer to [15].

The AudioSet ontology contains many sound classes that cannot be readily imitated vocally, such as *guitar amplifier* and labels related to music genres. After excluding these classes, 302 sound classes from the AudioSet ontology remained. AudioSet's actual audio typically contains scenes with multiple sounds, rather than isolated sounds. Since the goal of our data set is to provide clear pairings of vocal imitations to reference sounds, this makes AudioSet's audio sub-optimal. Therefore, we collected our sounds from a repository where contributors typically provide isolated, single-sound recordings. For each of the 302 selected sound classes, we collected an average of 10 audio recordings from *Freesound* using the class name as the search key. All files were truncated to a maximum of 20 seconds and encoded in the WAV format with a sample rate of either 44.1 kHz or 48 kHz.

A single high-quality recording was selected from the collected recordings for each class as a reference recording to be imitated by crowd-workers. Each reference audio file was confirmed to contain a clean sound event for the selected sound class and no other sound events. The other recordings that were not used for imitation collection are also included in the released dataset. Although they do not have associated vocal imitations, we expect that they will be useful for developing and evaluating fine-grained search algorithms (i.e., searching among sounds within the same class). We will show an example of using the recordings for fine-grained search in Section 4.

2.2. Vocal imitation collection

We collected vocal imitations from crowd-workers through Amazon Mechanical Turk using the VocalSketch interface and protocol presented in [17]. Imitators were asked to listen to a reference recording (i.e., one of the 302 collected reference recordings) and imitate the sound. Once they recorded their imitations, they were required to listen to their imitations to compare them with the reference recording. They were allowed to re-record their vocal imitations unlimited times before submitting the final one. Discarded imitations were saved as *draft recordings* in the released dataset. Finally, each imitator was asked how satisfied they were with their imitations using a 7 level scale. In each session, imitators were given five reference recordings (one recording from each class) to imitate. Imitators were paid \$0.80 per session. The first imitation of each imitator in a new session was saved as a *training recording*.

We collected a total of $11,242$ recordings from 455 unique people. There were $6,115$ final-submission vocal imitations, $4,444$ draft recordings and 683 training recordings. The $6,115$ final submission vocal limitations resulted in an average of roughly 20 imitations for each of the 302 reference recordings. We focused on this set of final submissions in our quality assessments.

3. QUALITY ASSESSMENT

Crowd-sourced data collection suffers from noisy data in many cases. Therefore, we conducted an internal quality assessment of the $6,115$ final submissions, where experts evaluated the quality of all the final collected imitations. Training and draft vocal recordings were not evaluated. The purposes of the quality assessment are the following: 1) removing non-identifiable vocal imitations from the data set, and 2) measuring perceptual similarity between a reference recording and its imitations. The people who performed quality assessment were experts in audio processing: students and researchers from the Interactive Audio Lab [3] at Northwestern University and the Audio Information Research Lab [4] at the University of Rochester. There were, in total 15 evaluators, who listened to $6,115$ vocal imitations on a web interface designed for this particular listening task.

Figure 1 shows the web interface for our quality assessment. A single session consists of listening to a pair of recordings: one reference and one vocal imitation (Sound A and Sound B in Figure 1). An evaluator was first asked if the imitation was a vocal imitation of the reference recording. If the answer was "YES", then the evaluator was asked to assess the quality of the imitation on a scale from 0 to 100 (0: a very poor imitation; 100: almost identical to the reference sound). If the answer was "NO", then the recording was not evaluated for quality and it was placed in the *excluded* directory of the released dataset. The evaluator was then asked if the recording was a vocal imitation at all and this answer was saved.

Due to the size of the dataset, each imitation was evaluated by a single person. To measure consistency and reliability of each evaluator, we designed the task in following ways. First, an average of 2 out of every 30 pairs evaluated by an individual were incorrect pairs, where we paired an imitation with a reference recording that

[3] http://music.cs.northwestern.edu/

[4] http://www.ece.rochester.edu/projects/air/

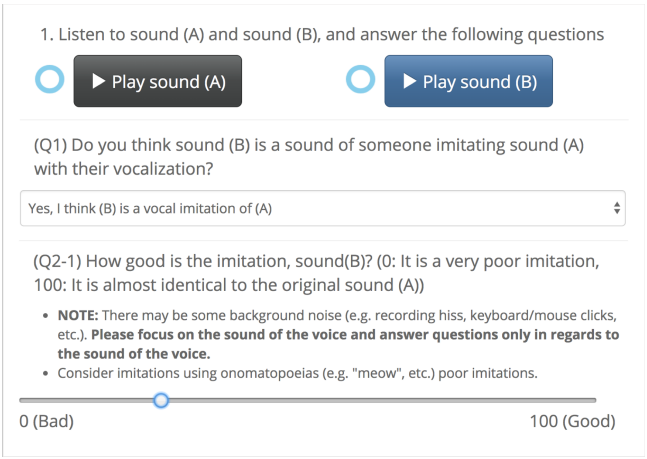


Figure 1: A screenshot of the interface for the internal quality assessment

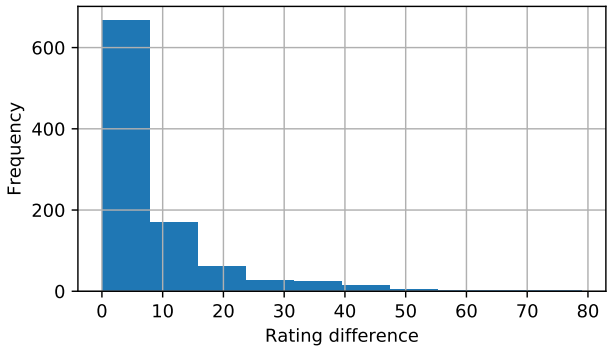


Figure 2: Histogram of maximum differences of quality ratings on a 100 point scale between two presentations of the same pairing of reference and imitation recording (Mean: 7.63, SD: 10.96)

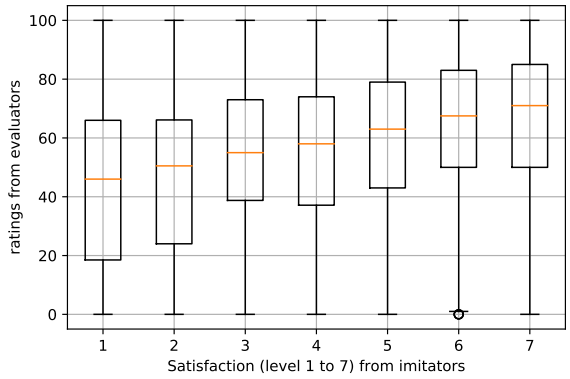


Figure 3: Relationship between self-satisfaction scores by imitators and quality assessment by evaluators.

it was not an imitation of. This let us measure how reliably evaluators were able to detect incorrect pairs. Second, an average of 4 out of every 30 pairs presented to an evaluator were repeated pairs, previously presented within the current batch (30 pairs). This let us measure the evaluation consistency for each evaluator.

In total, 452 incorrect pairs were presented to evaluators and 80% of them (363 pairs) were successfully identified as incorrect pairs. The remaining 20% (89 pairs) were incorrectly called correct pairs and they were given an average quality rating of 31.4 out of 100. The mean quality rating across all imitations is 60.3. This indicates that most evaluators correctly identified wrong pairs or gave them low scores if they called them a correct pair. Figure 2 shows how consistently evaluators rated repeated pairs. In total, 978 unique pairs of reference and imitation recordings were repeated. We computed the maximum difference of the multiple ratings to each of the 978 repeated pairs. For example, if a pair of reference and imitation recording was repeatedly rated three times by an evaluator and the ratings were 50, 60 and 70, then the maximum difference is 20 (70-50). As shown in Figure 2, the maximum differences of a majority of repeated pairs is very low (Mean: 7.63, SD: 10.96), which indicates that our evaluators rated vocal imitations with high consistency.

When collecting imitations, imitators were asked how satisfied they were with their own imitation using a 7 level scale. (1 - *completely dissatisfied,* 2 - *mostly dissatisfied,* 3 - *somewhat dissatisfied,* 4 - *neither satisfied or dissatisfied,* 5 - *somewhat satisfied,* 6 - *mostly satisfied,* 7 - *completely satisfied*). Figure 3 shows how the evaluator's ratings change with different self-satisfaction levels from imitators. There is a positive correlation between the imitators' self-satisfaction levels and evaluators' quality assessment scores. Yet, there are some imitations where the imitator's self-satisfaction disagrees with the quality reported by an evaluator. It would be interesting future work to learn the reason for the dichotomy.

Evaluators reported that 514 vocal imitations were not vocal imitation of the reference sound played to the imitator who made the imitation. These recordings were placed in the *excluded* directory of the released dataset. This left 5,601 recordings that have quality ratings, which are saved in the *included* directory of the dataset. We included all the quality rating on these 5,601 recordings in the released dataset.

4. BASELINE FINE-GRAINED SEARCH RESULTS

One expected use of Vocal Imitation Set is to train and test systems on fine-grained search. To this end, we provide an example of using this data set to measure how well the existing state-of-the-art in QBV search performs on fine-grained search. We used vocal imitations that were vetted by listeners (5,601 imitations of 302 classes). Each class contains one reference recording that was imitated and an average of 9 sound recordings that were not imitated. Each reference recording has an average of 18.6 imitations. Each time, we took one vocal imitation as the *query* to search for its reference recording (*target*) within all sound recordings of its class. An output of a search engine is an ordering of these sound recordings within each class, from most similar to the query to least similar.

To measure search quality, we computed Reciprocal Rank (RR), which is calculated as $1/r$ where r is the rank of the target. For instance, if the target ranks the third, then the RR is $1/3$. We measured *Mean Reciprocal Rank (MRR)*, which is the mean of RRs across all the queries (i.e., vocal imitations). We also computed a *Mean Recall@*k metric which indicates the proportion of queries that successfully retrieved the target within top k items in search results. For example, if only 50% of queries retrieved the target

recording within top 2 items, then *Mean Recall*@2 is 0.5.

We used TL-IMINET [12], which is the best system we are aware of for coarse-grained QBV retrieval. TL-IMINET is a Siamese-style neural network with two Convolutional Neural Networks (CNN) towers: one tower for vocal imitations and the other for reference recordings. The imitation tower was pre-trained on a language classification task using sound clips from 7 different languages gathered from Voxforge [5]. The reference tower was pre-trained on a environmental sound classification task using sound clips from the 10 different classes in UrbanSound8K [14]. Then, TL-IMINET was trained on positive and negative pairs of reference sounds and imitations from the VocalSketch dataset [17]. In this training, negative pairs were always from an entirely different sound class (e.g., an imitation of a dog bark with a reference recording of a door slamming), so these pairs can be considered coarse-grained.

We performed fine-grained search with the trained model as follows. The trained model outputs the similarity between two input recordings. An imitation as a query is compared to each audio file within the sound class of that imitation's canonical reference recording (i.e., target). Since TL-IMINET takes only four seconds of audio as an input, each audio file is segmented into windows of length four seconds, with 50% overlap between each window, which gives us segment-level cross-similarities between the two recordings. To obtain the recording-level similarity between the reference and query file, we took the maximum similarity between any two segments in the two recordings. Based on the similarities, the rank of the target within the class is determined. By running this search using every vetted vocal imitation (5,601 in total) as a query, we compute MRR as well as Mean Recall@k covering all classes and the variety of queries within each class.

TL-IMINET gave a MRR of 0.356 for within-class search. Mean Recall@1 was 0.151 and Mean Recall@2 was 0.278. The class with the best MRR was *"Water stream"* with MRR of 1.0 and the worst MRR was for *"Bird's chirp, tweet"* with a MRR of 0.105. Since the mean number of recordings per class is roughly 10, chance ranking of the target is 5.5 which leads to chance MRR of $(1/5.5) = 0.18$. The results show that the state-of-the-art system which were designed for coarse-grained search performs much better than chance. However, the score is still similar or lower than scores from coarse-grained search performed in [12]. They achieved a MRR of about 0.4 in searches through 20 recordings and a MRR of 0.246 in searches with 60 recordings. This comparison shows challenges of fine-grained search. We believe that Vocal Imitation Set will enable researchers to build and test new models for fine-grained QBV search.

5. VOCAL IMITATION SET

Vocal Imitation Set is now publicly available[6]. It includes 2,985 original recordings of 302 classes (an average of 9.89 per class) and 11,242 vocal imitations of 302 reference recordings selected from the set of original recordings (1 reference recording per class). The set of vocal imitations consists of 5,601 imitations that passed the quality assessment as well as 5,642 recordings of draft, training recordings, and imitations excluded during the quality assessment. Table 1 shows the number of classes, listener-vetted imitations (i.e., imitations that have quality ratings), and original recordings for each top-level classes of AudioSet ontology. Figure 4 shows a histogram of quality assessment ratings of the 5,601 listener-vetted imitations. The collected ratings give researchers another opportunity to build more robust vocal imitation-based interaction systems by using human quality assessments as a training signal.

[5] http://www.voxforge.org

[6] http://doi.org/10.5281/zenodo.1340763

Table 1: The number of classes, listener-vetted imitations, and original recordings (including reference recordings) for each of the first-level categories in Vocal Imitation Set

Categories	Classes	Imitations	Original Rec.
Animal	31	587	308
Channel, environment and background	4	74	40
Human sounds	38	714	375
Music	65	1247	646
Natural sounds	10	177	100
Sounds of things	134	2448	1316
Source-ambiguous sounds	20	354	200
Total	302	5601	2985

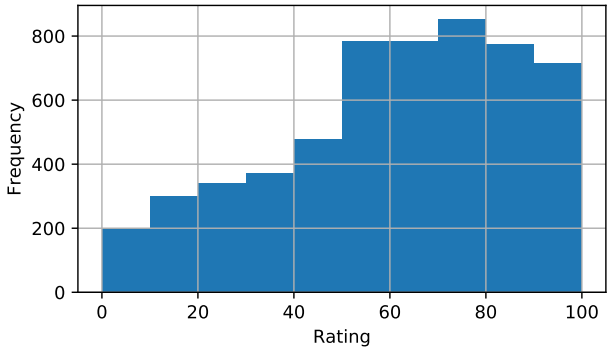


Figure 4: Histogram of quality assessment ratings to 5,601 vocal imitations that were vetted by evaluators (Mean: 60.3, SD: 25.3)

6. CONCLUSIONS

We introduced *Vocal Imitation Set*, a new dataset of vocal imitations. It contains 11,242 vocal imitations of 302 sound event classes which were curated based on AudioSet ontology. Sound recordings of the 302 classes were collected from Freesound and their imitations were collected by crowd-sourcing methods. We performed an internal quality assessment to filter out noisy data as well as to measure the perceptual similarity between an imitation and its reference recording. We also showed an example of using the dataset for fine-grained QBV search. We expect that this dataset will help the research community obtain a better understanding of human vocal imitations and build systems that can understand imitations as humans do.

7. REFERENCES

[1] G. Lemaitre and D. Rocchesso, "On the effectiveness of vocal imitations and verbal descriptions of sounds," *The Journal of*

the Acoustical Society of America, vol. 135, no. 2, pp. 862–873, 2014.

[2] G. Lemaitre, O. Houix, F. Voisin, N. Misdariis, and P. Susini, "Vocal imitations of non-vocal sounds," *PloS one*, vol. 11, no. 12, p. e0168167, 2016.

[3] D. Rocchesso, D. A. Mauro, and C. Drioli, "Organizing a sonic space through vocal imitations," *Journal of the Audio Engineering Society*, vol. 64, no. 7/8, pp. 474–483, 2016.

[4] O. Houix, S. D. Monache, H. Lachambre, F. Bevilacqua, D. Rocchesso, and G. Lemaitre, "Innovative tools for sound sketching combining vocalizations and gestures," in *Proceedings of the Audio Mostly 2016*. ACM, 2016, pp. 12–19.

[5] M. Cartwright and B. Pardo, "Synthassist: an audio synthesizer programmed with vocal imitation," in *Proceedings of the 22nd ACM international conference on Multimedia*. ACM, 2014, pp. 741–742.

[6] D. Rocchesso, G. Lemaitre, P. Susini, S. Ternström, and P. Boussard, "Sketching sound with voice and gesture," *interactions*, vol. 22, no. 1, pp. 38–41, 2015.

[7] A. Huq, M. Cartwright, and B. Pardo, "Crowdsourcing a real-world on-line query by humming system," in *Proceedings of the Sixth Sound and Music Computing Conference (SMC 2010)*, 2010.

[8] D. S. Blancas and J. Janer, "Sound retrieval from voice imitation queries in collaborative databases," in *AES 53rd Conference on Semantic Audio*, Audio Engineering Society. London, UK: Audio Engineering Society, 27/01/2014 2014.

[9] G. Roma and X. Serra, "Querying freesound with a microphone," in *Proceedings of the First Web Audio Conference (Ircam, Paris, France), submission*, vol. 39, 2015.

[10] Y. Zhang and Z. Duan, "Supervised and unsupervised sound retrieval by vocal imitation," *Journal of the Audio Engineering Society*, vol. 64, no. 7/8, pp. 533–543, 2016.

[11] S. Hershey, S. Chaudhuri, D. P. W. Ellis, J. F. Gemmeke, A. Jansen, C. Moore, M. Plakal, D. Platt, R. A. Saurous, B. Seybold, M. Slaney, R. Weiss, and K. Wilson, "Cnn architectures for large-scale audio classification," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017.

[12] Y. Zhang and Z. Duan, "Visualization and interpretation of siamese style convolutional neural networks for sound search by vocal imitation," in *Acoustics, Speech and Signal Processing (ICASSP), 2018 IEEE International Conference on*. IEEE, 2018.

[13] A. Mesaros, T. Heittola, A. Diment, B. Elizalde, A. Shah, E. Vincent, B. Raj, and T. Virtanen, "DCASE 2017 challenge setup: Tasks, datasets and baseline system," in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2017 Workshop (DCASE2017)*, November 2017, pp. 85–92.

[14] J. Salamon, C. Jacoby, and J. P. Bello, "A dataset and taxonomy for urban sound research," in *22nd ACM International Conference on Multimedia (ACM-MM'14)*, Orlando, FL, USA, Nov. 2014, pp. 1041–1044.

[15] J. F. Gemmeke, D. P. Ellis, D. Freedman, A. Jansen, W. Lawrence, R. C. Moore, M. Plakal, and M. Ritter, "Audio set: An ontology and human-labeled dataset for audio events," in *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*. IEEE, 2017, pp. 776–780.

[16] A. Mehrabi, S. Dixon, M. Sandler, *et al.*, "Towards a comprehensive dataset of vocal imitations of drum sounds," in *Proceedings of the 2nd AES Workshop on Intelligent Music Production*, 2016.

[17] M. Cartwright and B. Pardo, "Vocalsketch: Vocally imitating audio concepts," in *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. ACM, 2015, pp. 43–46.

[18] Y. Zhang and Z. Duan, "IMINET: Convolutional semi-siamese networks for sound search by vocal imitation," in *Applications of Signal Processing to Audio and Acoustics (WASPAA), 2017 IEEE Workshop on*. IEEE, 2017, pp. 304–308.

[19] A. Kumar, M. Khadkevich, and C. Fugen, "Knowledge transfer from weakly labeled audio using convolutional neural network for sound events and scenes," in *Acoustics, Speech and Signal Processing (ICASSP), 2018 IEEE International Conference on*. IEEE, 2018.

[20] Q. Kong, Y. Xu, W. Wang, and M. Plumbley, "Audio set classification with attention model: a probabilistic perspective," in *Acoustics, Speech and Signal Processing (ICASSP), 2018 IEEE International Conference on*. IEEE, 2018.

FAST MOSQUITO ACOUSTIC DETECTION WITH FIELD CUP RECORDINGS: AN INITIAL INVESTIGATION

Yunpeng Li[1], Ivan Kiskin[1], Marianne Sinka[2], Davide Zilli[1,3], Henry Chan[1], Eva Herreros-Moya[2], Theeraphap Chareonviriyaphap[4], Rungarun Tisgratog[4], Kathy Willis[2,5], Stephen Roberts[1,3]

[1] Machine Learning Research Group, Department of Engineering Science, University of Oxford, UK
{yli,ikiskin,dzilli,sjrob}@robots.ox.ac.uk, tsunhenry@gmail.com
[2] Department of Zoology, University of Oxford, UK
{marianne.sinka,eva.herreros-moya,kathy.willis}@zoo.ox.ac.uk
[3] Mind Foundry Ltd., UK
[4] Department of Entomology, Faculty of Agriculture, Kasetsart University, Bangkok, Thailand
aasthc@ku.ac.th, rungarun_tis@hotmail.com
[5] Royal Botanic Gardens, Kew, UK

ABSTRACT

In terms of vectoring disease, mosquitoes are the world's deadliest. A fast and efficient mosquito survey tool is crucial for vectored disease intervention programmes to reduce mosquito-induced deaths. Standard mosquito sampling techniques, such as human landing catches, are time consuming, expensive and can put the collectors at risk of diseases. Mosquito acoustic detection aims to provide a cost-effective automated detection tool, based on mosquitoes' characteristic flight tones. We propose a simple, yet highly effective, classification pipeline based on the mel-frequency spectrum allied with convolutional neural networks. This detection pipeline is computationally efficient in not only detecting mosquitoes, but also in classifying species. Many previous assessments of mosquito acoustic detection techniques have relied only upon lab recordings of mosquito colonies. We illustrate in this paper our proposed algorithm's performance over an extensive dataset, consisting of cup recordings of more than 1000 mosquito individuals from 6 species captured in field studies in Thailand.

***Index Terms*—** Mosquito detection, acoustic signal processing, multi-species classification, convolutional neural networks

1. INTRODUCTION

Malaria results in half a million deaths each year and mosquitoes are the only vector for malaria [1]. Among more than 3500 mosquito species, only around 60 out of the 450 *Anopheles* species can transmit malaria parasites to infect humans, i.e. are vectors [2]. Therefore, detailed mosquito surveying in areas of endemic malaria is crucial to identify the distribution of malaria-vectoring mosquitoes.

Standard mosquito sampling approaches, including human landing catches, odour-baited traps and cow-baited tents, can be effective in sampling malaria vectors [3, 4]. However, they expose volunteers to potentially infectious bites or are not sufficiently efficient for large-scale and frequent monitoring of mosquito distributions. An alternative solution, using mosquito flight tones to distinguish species, has been researched for some 60 years [5, 6]. In recent years, proof-of-concept mosquito acoustic sensing paradigms, based on embedded devices such as mobile phones, have been proposed [7, 8, 9]. Embedded devices provide a compelling platform for such environmental acoustic sensing tasks due to their cheap and efficient sensors, wide availability and built-in storage and wireless connectivity [10].

Research in the signal processing aspect of mosquito acoustic sensing has often focused on two areas. Firstly the use of domain knowledge to extract hand-crafted features to then allow high-quality detections and secondly the construction of machine learning frameworks which are well-suited to not just detect mosquitoes but importantly also to distinguish species. In much work, fundamental frequencies and associated harmonics form the basis for models which identify mosquito species [11, 9]. However, these low-dimensional features suffer from high intra-species variances and significant overlaps between different species [11, 12], hence limiting their application in multi-species classification. Alternative approaches look to avoid such feature construction and instead allow machine learning algorithms to extract relevant information direct from e.g. the spectrogram. Promising detection results have been reported [8, 13], though we note that the datasets used in evaluations of most previous work are limited in their sample sizes and were usually collected with mosquitoes raised in lab environments.

As a part of the HumBug project[1], a two-month mosquito survey was conducted in rural Thailand. A total of 1256 individual mosquitoes of 9 different mosquito species were captured and the flight tones of these mosquitoes were recorded for each captured individual. We here present the development of a machine learning algorithm that is computationally efficient (as it needs to be for implementation on low-powered embedded devices) and report in this paper on its performance over this field-recorded dataset.

The rest of this paper is organised as follows. We describe in Section 2 the dataset and the proposed mosquito acoustic detection algorithm. In Section 3 we report detection performance and discuss results. We conclude the paper and discuss future directions in Section 4.

[1] humbug.ac.uk

2. DATA AND METHODS

2.1. Field experiments and data summary

A two-month comprehensive survey of mosquito fauna was conducted at Pu Teuy Village, Sai Yok District, Kanchanaburi Province, Thailand. The survey was conducted within the peak mosquito season (May to October), and ran from the 12th of June until the end of July in 2018. Three methods of capture were used: human baited nets (HBN), cow baited nets (CBN) and larval collections. The HBN and CBN were run for 12 hours over night with collections made each hour throughout. All adult mosquitoes were placed into sample cups large enough for them to fly freely and their flight was recorded the morning following capture.

Recording was conducted using two microphone set-ups: 'budget' and 'high spec'. The 'budget' set-up used an Alcatel One-Touch Pixi smartphone with a TIE 19-90003 condensor microphone. The budget setup also used our 'Mozzwear' app on Alcatel smartphones to perform data capture and digitisation. The 'high-spec' set-up used a high specification field microphone (Telinga EM-23) plugged into a digital sound recorder (Olympus LS-14). Monophonic recordings were collected in both set-ups. Larval collections were made along a small river with known anopheline larval sites and the larvae/pupae were placed into rearing trays. The emerging adults were placed into individual sample cups and provided with $10\% w/v$ sucrose solution and recorded as above. As of the 20th July 2018, a total of 1256 individual mosquitoes had been captured. The detailed number of individuals for each species is reported in Table 1. A total of 127 mosquito individuals died before recording, 92 were lost, 46 did not fly and 21 individuals are as yet unidentified.

After recording the mosquito flight tones of these captured mosquito individuals, data tagging was required to mark segments of recordings with mosquito flight tones. Our project research team labelled a subset of the recordings and obtained more than 1 hour of mosquito flight tones from the field-captured mosquitoes, in addition to background recordings. The number of mosquito individuals and durations of flight tones of each species are shown in Table 1.

Table 1: Number of captured mosquito individuals and durations of recorded mosquito flight tones for different species.

Mosquito species	# individuals	Recorded time
Aedes sp.	256	954 seconds
An. maculatus	105	486 seconds
An. dirus	110	474 seconds
An. harrisoni	150	612 seconds
Armigeres sp.	261	1084 seconds
Culex sp.	67	386 seconds
Mansonia sp.	4	14 seconds
An. minimus	8	61 seconds
An. barbirostris	9	13 seconds

2.2. Mel-frequency spectrum-based convolutional neural networks

In this paper we propose a computationally efficient multi-species classification pipeline. As our goal is to port the model onto limited resource hardware, we base the feature encoding on the mel-frequency spectrum (MFS) and use convolutional neural networks (CNNs) as the decision engine. The mel-frequency is chosen due to its effectiveness as well as its interpretability. Although the CNN incurs well-known computational costs during training, running the trained CNN on new data is efficient. We detail below the feature encoding approach taken as well as the classification algorithm.

2.2.1. Feature representation

Previous work [8, 13] identified either mel-frequency cepstral coefficients (MFCCs) or wavelets as effective features for multiple machine learning algorithms. MFCCs are computationally efficient and have been a popular choice in mosquito detection and other acoustic scene classification tasks [8, 10]. However, the discrete cosine transform (DCT) step leads to less human-interpretable features than MFCCs, as shown in Figure 1, where the mel-frequency spectrum (Figure 1(d)) better preserves the harmonic structure in the spectrogram (Figure 1(b)) in comparison to the MFCC (Figure 1(c)). Further, the mel-frequency spectrum is also fast to compute and it forms a compact representation of the spectrum. Our experiments, including that presented in Section 3, have shown that the mel-frequency spectrum leads to detection performance with no statistically significant difference to results obtained with the MFCC.

We therefore use the mel-frequency spectrum to construct time-frequency representations of audio recordings in the same spirit of the short-time Fourier transformation (STFT). For a short audio clip, e.g. 0.1 second, we can compute the mel-frequency spectrum for smaller segments within the clip, and combine them to form a time-frequency matrix. This resultant matrix forms the input space of the subsequent machine learning algorithm (Figure 2).

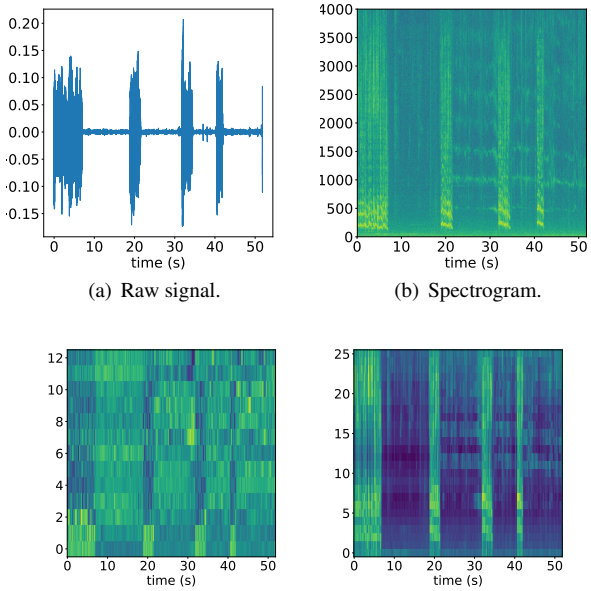
Compared with the spectrogram or the wavelet, this mel-frequency, spectrum-based representation has much smaller dimension - making it efficient for model training with datasets of small sample sizes, such as those often encountered in our application. We note that the wavelet transformation has been shown to provide informative time-frequency features that are well-suited to subsequent use, particularly by convolutional neural networks (CNNs) [13]. However, the wavelet transformation used in this latter work is computationally demanding and unsuited for the task of real-time mosquito species classification on low-cost phones.

2.2.2. Classification algorithm

Convolutional neural networks (CNNs) are a subset of (deep) neural networks that have been widely used in processing two-dimensional inputs, e.g. images [14, 15]. The first layer in a CNN consists of a set of convolutional filters, whose parameters (the filter coefficients) are learned as part of the training process. These filters act so as to create latent representations of the observations which are passed upwards to layers in the CNN which create discriminants associated with the classes of interest. This approach, crucially, does not prespecify the form of patterns in the data which are highly informative. CNNs have been widely used in the machine vision literature and we exploit their excellent performance over images by treating the mel-frequency spectrum as a 2d image.

The convolutional layer takes an input tensor $X \in \mathbb{R}^{h_1 \times w_1 \times c}$ where $c = 1$ for the mel-frequency spectrum-based features. N_k kernels $K_p \in \mathbb{R}^{k,k}$ for $p \in \{1, \ldots, N_k\}$ are applied to the input tensor which produces the convolution output Y_p:

$$Y_p(i,j) = (X * K)(i,j) = \sum_m \sum_n X(i-m, j-n) K(m,n) \ . \quad (1)$$



(a) Raw signal. (b) Spectrogram.

(c) MFCC (plotted over time). (d) Mel-frequency spectrum (plotted over time).

Figure 1: An example recording and associated spectral features. Mosquito flight tones are present from 21 s to 31 s for cow capture #242, 35 s to 40 s from cow capture #243, 42 s to 52 s from cow capture #251. Other segments of the recording (including the high amplitude sections) are either background noise or human voices.

These feature maps, i.e. convolution outputs, are then passed through some non-linear activation function, such as a Rectified Linear Unit (ReLU), before being flattened and connected to fully connected hidden layers. The last layer consists of N nodes where N is the number of classes.

2.3. Training strategy

In training and evaluating mosquito acoustic detection algorithms, we first randomly remove 50% of recordings described in Section 2.1 to create the hold-out test dataset. For the remaining recordings, we divide recordings into audio clips with a length of 0.1 seconds, thus creating a relatively large number of samples with which we train the CNN and other benchmark algorithms.

As shown in Table 1, the dataset is highly imbalanced. To avoid issues with very small data sample sizes, we only use samples from the *Aedes* sp., *An. Maculatus*, *An. dirus*, *An. harrisoni*, *Armigeres* and *Culex* sp. to evaluate the detection performance of the algorithms, as there is less than 2 minutes of recordings for the other species. Three of these species are known malaria vectors, including *An. maculatus*, *An. dirus* and *An. harrisoni*. Following [8], we randomly sample the training samples, without replacement, to produce a balanced training set. In our application, this creates a data set of close to 2000 samples for each mosquito species. A total of 100 randomised trials were performed so that different training and test data sets were produced among different simulation trials.

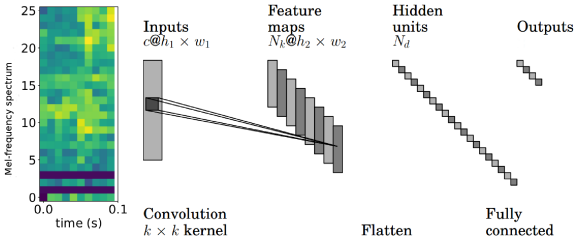


Figure 2: Example CNN architecture. Input to the CNN is a mel-frequency spectrum computed from a 0.1 second audio clip with $c = 1$ channel and dimension $h_1 \times w_1$. The CNN has N_k filters with kernel $K \in \mathbb{R}^{k \times k}$ thus it reduces the input dimension to $h_2 \times w_2$ following convolution with each filter. These feature maps are flattened (i.e. vectorised) before being fully connected to N_d hidden units in a dense layer. The last fully-connected layer produces the classification output.

3. EXPERIMENT RESULTS

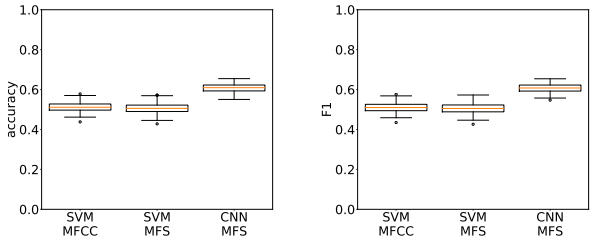
3.1. Parameter setup and benchmarking

The input image to the CNN is of dimension 26×10, where 26 is the dimension of the mel-spectrum and 10 is the number of 0.01 second windows within a 0.1 second audio clip. In this initial investigation we adopt a network structure inspired by [16] which was used for MNIST handwriting digit recognition. There are three convolutional layers: the first layer consists of 8 filters, the second one has 32 filters, and the last one is made up of 64 filters. All filters have 3×3 kernel size. They are followed with one hidden layer of 128 nodes. The ReLU activation unit and a dropout rate of 0.3 are used. The neural network is trained using the Adam algorithm [17] with a batch size of 256 for 100 epochs. A full cross-validation of these default parameter values, and the optimisation of network structure, will be performed in follow-up studies.

We choose as benchmark classifier a support vector machine (SVM) using a one-versus-one multi-class classification strategy [18]. The one-versus-one multi-class classification strategy has a simpler data balancing requirement compared to one-versus-rest. As discussed in [8], MFCC features combined with a SVM obtains the best multi-species classification accuracy among several common acoustic features and off-the-shelf detection algorithms. However, subsequent studies showed that the mel-frequency spectrum leads to similar detection performance and more human-interpretable features.

3.2. Results

Figure 3 plots the out-of-sample classification accuracy and F1 score of the compared algorithms, respectively. The mel-frequency spectrum (MFS)-based CNN algorithm exhibits significantly better classification performance, in terms of both classification accuracy and F1 score over the SVM algorithms. We observe no significant difference between results obtained with MFCCs and the mel-frequency spectrum. As shown in Figure 1, the mel-frequency spectrum is able to better preserve the harmonic structure of the mosquito flight tone than MFCCs. Figure 4 and 5 plot confusion matrices for the MFS-based SVM and the MFS-based CNN, respectively. The CNN exhibits better mean sensitivities than the SVM for every mosquito species in this experiment.



(a) Classification accuracy. (b) F1 score.

Figure 3: Boxplots showing out-of-sample classification accuracy and F1 scores across 100 randomised trials.

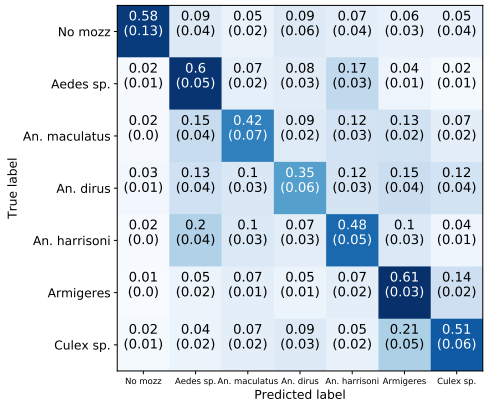


Figure 4: Confusion matrix of out-of-sample classification performance for the mel-frequency spectrum-based SVM. The first value in each entry is the corresponding mean value among 100 simulation trials, while the value in the parenthesis reports the standard deviation

Considering the fact that the dataset contains recordings from more than 1000 mosquito individuals from 6 species, an average classification accuracy of above 60% from the CNN (Figures 3 and 5), using half of the samples for training, is very encouraging. Once trained, both the SVM and the CNN are computationally efficient in their prediction step, allowing their real-time execution in low-cost low-power embedded devices.

Compared to the classification results reported in [8], where an average of 80% classification accuracy is achieved with a SVM, we notice a significant decrease of classification accuracy with this 'in the wild' dataset. It is important to note that the data size is significantly smaller in [8] where only 6.2 seconds of recordings were available for each species after data resampling, and only one mosquito individual per species was used to collect recordings. This suggests higher correlations between different samples in the lab-collected dataset of [8] and highlights the importance of performance evaluation with large-scale mosquito flight tone datasets.

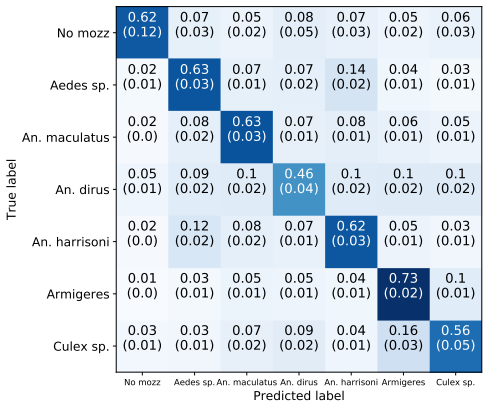


Figure 5: Confusion matrix of out-of-sample classification performance for the mel-frequency spectrum-based CNN. The first value in each entry is the corresponding mean value among 100 simulation trials, while the value in the parenthesis reports the standard deviation

4. CONCLUSION

Mosquito acoustic detection aims to provide an alternative solution to fast sample and update mosquito species distribution, which is crucial for control programmes and guiding intervention policies. The HumBug project uses mobile phones for mosquito acoustic detection and species classification. Our dataset, recently collected in field sites in Thailand, provides a valuable resource to develop and evaluate mosquito acoustic detection algorithms. The nature of the task requires an algorithm with a low computational cost while maintaining effectiveness with a small number of training samples.

We propose in this paper a computationally efficient classification pipeline, based on the mel-frequency spectrum and convolutional neural networks. The mel-frequency spectrum (MLS) is a computationally efficient, low-dimensional acoustic feature. We show that the proposed pipeline, with the CNN acting to construct latent features from the MLS, achieves impressive classification performance on a challenging field dataset.

This initial investigation reports classification results with labelled data from a subset of recordings in which labels were obtained by data tagging from project team members. Further work will include data which is being labelled via citizen scientists on the Zooniverse[2] citizen science platform. Working with such data will require for algorithms capable of handling crowdsourced labels which are at different resolutions to the original data frames - and this is a topic of active current research. Further directions also include optimisation of the network structure, identification of more effective feature extraction methods, as well as the incorporation of the mosquito field dataset with other DCASE Challenge datasets to form a large-scale acoustic sensing task in future DCASE Challenge events.

ACKNOWLEDGEMENTS

This work is part-funded by a Google Impact Challenge award.

[2]https://www.zooniverse.org/projects/yli/humbug

5. REFERENCES

[1] S. Bhatt et al., "The effect of malaria control on Plasmodium falciparum in Africa between 2000 and 2015," *Nature*, vol. 526, no. 7572, pp. 207–211, 2015.

[2] D. E. Neafsey et al., "Highly evolvable malaria vectors: The genomes of 16 anopheles mosquitoes," *Science*, vol. 347, no. 6217, 2015.

[3] J. Wong et. al., "Standardizing operational vector sampling techniques for measuring malaria transmission intensity: evaluation of six mosquito collection methods in western Kenya," *Malaria Journal*, vol. 12, no. 1, p. 143, Apr 2013.

[4] B. St. Laurent et. al., "Cow-baited tents are highly effective in sampling diverse anopheles malaria vectors in cambodia," *Malaria Journal*, vol. 15, no. 1, p. 440, Aug 2016.

[5] W. H. O. Jr. and M. C. Kahn, "The sounds of disease-carrying mosquitoes," *The Journal of the Acoustical Society of America*, vol. 21, pp. 462 – 463, 1949.

[6] C. Pennetier, B. Warren, K. R. Dabir, I. J. Russell, and G. Gibson, "singing on the wing as a mechanism for species recognition in the malarial mosquito Anopheles gambiae," *Current Biology*, vol. 20, no. 2, pp. 131 – 136, 2010.

[7] I. Kiskin, B. P. Orozco, T. Windebank, D. Zilli, M. Sinka, K. Willis, and S. Roberts, "Mosquito detection with neural networks: the buzz of deep learning," *arXiv:1705.05180*, 2017.

[8] Y. Li, D. Zilli, H. Chan, I. Kiskin, M. Sinka, S. Roberts, and K. Willis, "Mosquito detection with low-cost smartphones: data acquisition for malaria research," in *NIPS Workshop on Machine Learning for the Developing World*, Long Beach, USA, Dec. 2017, arXiv:1711.06346.

[9] H. Mukundarajan, F. J. H. Hol, E. A. Castillo, C. Newby, and M. Prakash, "Using mobile phones as acoustic sensors for high-throughput mosquito surveillance," *eLife*, vol. 6, p. e27854, Oct. 2017.

[10] D. Barchiesi, D. Giannoulis, D. Stowell, and M. D. Plumbley, "Acoustic scene classification: Classifying environments from the sounds they produce," *IEEE Signal Processing Mag.*, vol. 32, no. 3, pp. 16–34, 2015.

[11] I. Potamitis and I. Rigakis, "Measuring the fundamental frequency and the harmonic properties of the wingbeat of a large number of mosquitoes in flight using 2d optoacoustic sensors," *Applied Acoustics*, vol. 109, pp. 54 – 60, 2016.

[12] S. M. Villarreal, O. Winokur, and L. Harrington, "The impact of temperature and body size on fundamental flight tone variation in the mosquito vector Aedes aegypti (diptera: Culicidae): Implications for acoustic lures," *Journal of Medical Entomology*, vol. 54, no. 5, pp. 1116–1121, 2017.

[13] I. Kiskin, D. Zilli, Y. Li, M. Sinka, K. Willis, and S. Roberts, "Bioacoustic detection with wavelet-conditioned convolutional neural networks," *Neural Computing and Applications*, 2018, accepted.

[14] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural Computation*, vol. 1, no. 4, pp. 541–551, 1989.

[15] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. Int. Conf. Neural Information Processing Systems*, Lake Tahoe, USA, Dec. 2012, pp. 1097–1105.

[16] P. Y. Simard, D. Steinkraus, and J. Platt, "Best practices for convolutional neural networks applied to visual document analysis," in *Proc. Int. Conf. Document Analysis and Recog. (ICDAR)*, Washington, DC, USA, Aug. 2003.

[17] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv:1412.6980*, 2014.

[18] C. Cortes and V. Vapnik, "Support-vector networks," *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, Sept. 1995.

USING AN EVOLUTIONARY APPROACH TO EXPLORE CONVOLUTIONAL NEURAL NETWORKS FOR ACOUSTIC SCENE CLASSIFICATION

Christian Roletscheck, Tobias Watzka, Andreas Seiderer, Dominik Schiller, Elisabeth André

Augsburg University
Human Centered Multimedia
Augsburg, 86159, Germany
rolle.roletscheck@t-online.de,tobias.watzka@gmail.com,{seiderer, schiller, andre}@hcm-lab.de

ABSTRACT

The successful application of modern deep neural networks is heavily reliant on the chosen architecture and the selection of the appropriate hyperparameters. Due to the large number of parameters and the complex inner workings of a neural network, finding a suitable configuration for a respective problem turns out to be a rather complex task for a human. In this paper we, propose an evolutionary approach to automatically generate a suitable neural network architecture and hyperparameters for any given classification problem. A genetic algorithm is used to generate and evaluate a variety of deep convolutional networks. We take the DCASE 2018 Challenge as an opportunity to evaluate our algorithm on the task of acoustic scene classification. The best accuracy achieved by our approach was 74.7% on the development dataset.

Index Terms— Evolutionary algorithm, genetic algorithm, convolutional neural networks, acoustic scene classification

1. INTRODUCTION

Deep neural networks (DNNs) have already proven their capability to achieve outstanding performance in solving various classification tasks. Therefore, it is reasonable to use them for acoustic scene classification as well. This trend can be clearly seen in the DCASE Challenge of 2016 [1] and 2017 [2]. However, designing the network architecture and finding the corresponding hyperparameters (learning rates, batch size etc.) remains a challenging and tedious task, even for experts. Therefore, a lot of attention in recent research has been payed on finding ways to automate this process [3, 4, 5, 6]. Among others the *neuro-evolution* method, which relies on evolutionary algorithms (EAs), has been a prominent choice for the task of automatically generating neural networks (NNs). In this work we are exploring the capabilities of neuro-evolution to discover an optimal DNN topology and its hyperparameters for the task of acoustic scene classification. Furthermore, we are introducing our novel self-adaptive EA which uses a genetic representation to create DNNs: *Deep Self-Adaptive-Genetic-Algorithm* (*DeepSAGA*).

2. RELATED WORK

One of the earliest works using EAs to generate NNs, was conducted by Miller et al. [7]. While their approach was originally limited to only evolve the weights of the NN they also showed that it could be advantageous to use an EA to generate a complete NN architecture. In the year 2002 Stanley and Miikkulainen introduced a method called *NeuroEvolution of Augmenting Topologies* (NEAT) which evolves NN topologies along with the weights [8]. Till today NEAT is the basis for many state-of-the-art algorithms in the field of neuro-evolution.

Kroos and Plumbley proposed 2017 in [9] a modified version of the NEAT algorithm. Their EA "J-NEAT" generates small NNs for sound event detection in real life audio. As participants of the DCASE Challenge 2017 they demonstrated that their generated small NNs are able to compete with other bigger networks, such as the baseline approach. Their main concern, however, was the minimization of the total number of nodes used in the generated NN rather than the maximization of the classification performance.

Real et al. [10] have shown in 2017 the capability of EAs to create CNNs solving image recognition tasks. Their fully generated models are capable of competing with the state-of-the-art models manually created by experts. This boost in classification performance, however, comes at the expenses of computational costs. The discovery of the best model took a wall time of roughly 256 hours of evolutionary search, distributed over 250 worker clients. While this approach, therefore, demonstrates the general feasibility of utilizing EAs to discover new DNN architectures it is also largely incapable for most practical applications due to its extensive demand of computation power.

Martin et al. [11] developed a novel EA that evolves the parameters and the architecture of a NN in order to maximize its classification accuracy, as well as maintaining a valid sequence of layers. Parameters related to their EA where empirically chosen by hand and won't change during a run.

The shown state-of-the-art approaches have demonstrated the general feasibility of using EAs to discover new DNN topologies and hyperparameters. However, given their respective specific characteristics, none of them seems to be an optimal fit in our case. Like the other algorithms mentioned here, DeepSAGA evolves the architecture and hyperparameters of a NN as well, while using the backpropagation algorithm [12] for weight optimization. Its main goal lies on the maximization of the classification performance for a given classification problem with a limited amount of disposable compute power. In addition, its own parameters are included in the evolutionary search process, making the search in advance for optimal start parameters obsolete and giving the algorithm the chance to change its parameters autonomously during a run.

3. DEEPSAGA

For the development of our approach, we followed the guidelines set forth by Eiben et al. [13] for designing executable evolutionary algorithms. The creation of an executable EA instance requires

the specification of its parameters. One of their guidelines suggests using parameter control [13, Chap. 7.3] for finding suitable parameters easier, since the resulting values not only influence the finding of an optimal solution, but also the efficiency. In our case, we use the *Self-Adaptive* variant, as it is one of the possible *Parameter-Control* techniques. In this variant the parameters to be adapted are represented as a component of the genetics and are thus part of the evolutionary search space. Therefore, has the potential to adapt the algorithm to the problem while solving it [13, Chap. 8].

The following subsections describe the implementation of our EA, while still taking the guidelines by Eiben et al. into account. The representation and definition of our individuals, the used fitness function and details to our population will be explained in the subsections 3.1, 3.2 and 3.3 respectively. Other subsections in this section are related to the typical steps, that will be excluded by an EA during its run. In the entire course of this work, the term session refers to the holistic process of an EA (from initialization to termination), while the repetition of the steps, selection of parents, recombination, mutation, evaluation of offspring and selection of individuals to form the next population are called a cycle.

3.1. Representation and definition

To enhance readability, we have used a representation analogous to biological genetics. Within biological terminology, a **genome** contains all **chromosomes** and represents the entire genetic of a living being. A chromosome is a bundle of several **genes** contained in the organism, whereby genes determine the different characteristics of that organism.

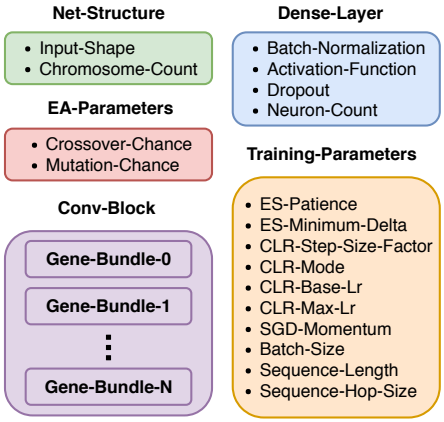


Figure 1: Detailed overview of all chromosomes and genes listed in a genome. ES: early stopping; CLR: cyclic-learning-rate.

In our work, a genome represents the genetic of a NN and describes its characteristics: its architecture and hyperparameters. The **genotype** is expressed by our genome and the **decoding**, in our case, the process of creating and training the network according to the characteristics described by the genotype.

Figure 1 lists our chromosomes contained in each genome. The **Conv-Block** is made up of at least one so-called **Gene-Bundle**. For each Gene-Bundle a convolutional layer followed by a max-pooling layer will be added to a NN architecture. It therefore contains information (genes) about the number of filters, filter-shape and filter-stride. In addition, each Gene-Bundle contains genes indicating whether optional layers for zero-padding, dropout and batch-normalization are included. Finally, a global-average-pooling or flatten layer can be used to connect the Conv-Block with the output or further classification layers.

An **allele** is a concrete expression of a gene. In our chosen representation model, an allele for the **Batch-Size-Gene** could be, for example, the integer number 128. Finally, Figure 2 illustrates the overall design of our genome. An example for a genome with filled in values can be seen in Figure 4, while Figure 5 displays said genome generated CNN architecture.

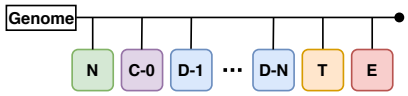


Figure 2: Illustration of a genome. The squarelike objects are representatives for the chromosomes Net-Structure, Conv-Block, Dense-Layer, Training- and EA-Parameters.

3.2. Fitness function

In our case, the focus was mainly on the accuracy of a NN. However, to speed up the evolutionary search process, the number of training epochs of a network was also taken into account. As a result, our $score$ value represents the total fitness of a population member, meaning the higher the $score$ the better the quality of a genotype. The following formula illustrates the utilized fitness function:

$$\mathbf{score} = 0.98 * accuracy + 0.02 * \frac{epoch_{limit} - epoch}{epoch_{limit}} \quad (1)$$

In this context, $epoch_{limit}$ stands for the maximum number of epochs a net is permitted as training time and $epoch$ for the number of epochs with which the net was actually trained. The distribution with 98 % on accuracy and 2 % on the other half, seems to be a solid approach and is solely based on own empirical observations.

3.3. Population

A steady state model [13, Chap. 5.1] is used, to manage the population. To promote diversity and the self-adaptive property, the population size is dynamic. However, since the available resources are limited, the maximum population size is restricted to 90.

3.4. Parent selection mechanism

As described by Bäck and Eiben [14] the parents are determined by a tournament selection procedure. The tournament depends on the population of the current cycle. This also applies to the number of population members who are allowed to participate in the tournament. To calculate said number Formula (2) was used, which takes into account the maximum permitted population size.

$$\mathbf{participants} = popsize_{limit} - popsize_{current} \quad (2)$$

The tournament size is determined by Formula (3), where $toursize_{limit}$ always corresponds to one tenth of the $popsize_{limit}$.

$$\mathbf{toursize} = toursize_{limit} * \frac{popsize_{current}}{popsize_{limit}} \quad (3)$$

If the population limit is reached, the number of participants is limited to **two** until the threshold value is undershot again.

3.5. Variation operators

Mutation

Genes of the category symbolic are mutated by replacing the original allele with a randomly selected. However, the **current** allele has a higher chance of being selected again than the other possible alleles. This type of mutation is also called sampling.

An allele of the integer type is mutated by a creep mutation [13, Chap. 4.3.1] or reset, whose chance is set to 5 %. In our case the sigma value always corresponds to 0.025 (2.5 %) times the limit. For example, if the maximum limit is 1000, the corresponding sigma value would be 25.

Nonuniform mutation [13, Chap. 4.4.1] is used to mutate an allele of the float type. This time, the sigma value is equated with the individual's chance of mutation.

Each population member has its own chance of mutation, which is co-evolved according to the method described in [15]. Before all other genes, the **Mutation-Chance-Gene** is mutated using the nonuniform mutation method. The resulting new mutation chance is the probability with which the remaining genes are mutated.

Recombination

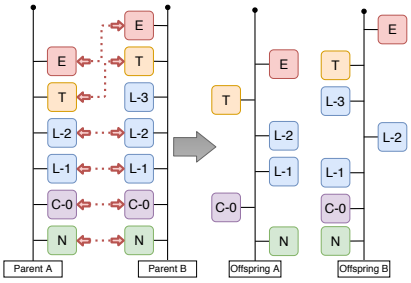


Figure 3: Illustrated procedure of our uniform crossover

The probability with which a recombination (throughout this work referred to as crossover) takes place depends on the crossover chance. Since, in our work, the crossover chance is part of the evolutionary process, it is represented by the **Crossover-Chance-Gene**. Thus, each population member has an individual crossover chance. The recombination process is based on the procedure described in [13, Chap. 8.4.7]. The individual crossover chance p_c of a parent is compared with a random number r ($r \in [0, 1]$). One parent is "ready to mate" if $p_c > r$ applies. This opens up the following possibilities:

1. When both parents are ready to mate, a crossover takes place.
2. If both parents are not ready to mate, they are cloned.
3. If only one parent is willing to mate, a clone of the unwilling parent is created. For the remaining individual a new partner is chosen randomly from the pool of parents, who is also checked for its willingness to mate.

The recombination itself takes place in the style of a uniform crossover [13, Chap. 4.2.2]. For example, offspring A first receives all chromosomes of parent A, then each of the chromosomes of offspring A could be swapped with a chromosome of parent B, taking the crossover chance of parent A into account. The exact procedure is depicted in Figure 3.

3.6. Survivor selection mechanism

Our selection procedure follows an age-based [13, Cap. 5.3.1] replacement strategy. Thus, each newly created individual is assigned a value (remaining lifetime, in short **RLT**) using Formula (4) as described by Bäck [14]. The RLT is reduced by **1** after each cycle, thus determining how long a population member remains alive. Though, the lifetime of the individual with the highest fitness remains unchanged. Where $MinLT\ (\alpha)$ and $MaxLT\ (\omega)$ stand for the permissible minimum and maximum lifetime of an individual. All other variables are linked to the current status of the population. These variables are $fitness\,(i)$, $AvgFit\ (AF)$, $BestFit\ (BF)$ and $WorstFit\ (WF)$. They stand for the fitness of the individual i, the average fitness, the best fitness and the worst fitness of the current population. The prefactor calculation is $\eta = \frac{1}{2} \cdot (\omega - \alpha)$.

$$\mathbf{RLT}\,(i) = \begin{cases} \alpha + \eta \cdot \frac{WF - fitness(i)}{WF - AF} & \text{if } fitness\,(i) \geq AF \\ \frac{1}{2}\,(\alpha + \omega) + \eta \cdot \frac{AF - fitness(i)}{AF - BF} & \text{if } fitness\,(i) < AF \end{cases} \tag{4}$$

The authorized minimum and maximum lifetime of an individual has been set to **1** and **7**. If the fitness value of a newly created individual i is better than the average fitness, it receives a lifetime from **5** to **7**, otherwise a lifetime from **1** to **4**. Within these subareas, the better individuals have a longer lifespan than the individuals with a lower fitness.

3.7. Initialization and termination

The individuals of the first population are generated randomly. Since the available resources are limited, the expressions of the respective genes are bounded. These limits must not be exceeded by variation operators either. The termination criterion is the completion of the **40th** cycle, considering the optimum is, in our case, not known in advance.

4. EXPERIMENTS

4.1. Setup

To evaluate the proposed genetic algorithm we use the TUT Urban Acoustic Scenes 2018 dataset from subtask A provided by the DCASE 2018 Challenge [16]. The dataset consists of 10-seconds audio segments from 10 different acoustic scenes. For every acoustic scene, audio was captured in 6 different cities and multiple locations. To train and measure the performance of the generated models we use the development dataset with the suggested partitioning for training and testing.

To generate the input features for the NNs the stereo audio samples were first converted into mono channels. Thereafter, the librosa library (v0.6.1) [17] was used to extract log mel spectrograms with 100 mel bands that cover a frequency range of up to 22050 Hz. For the Short-Time Fourier Transform (STFT) a Hamming window with a size of 2048 samples (43 ms) and a hop size of 1024 samples (21 ms) was used. The resulting spectrograms were divided into sequences with a certain number of frames defining the sequence length. For the creation of the sequences an overlap of 50 % was used. The sequence length can vary depending on the different models generated by the genetic algorithm.

Due to the stochastic behaviour of the algorithm, two independent sessions of 10 cycles each were initially completed. After-

wards, the best 30 models from each of these sessions were added to the initial population of a final third session. This approach results in a population with a higher initial fitness while keeping a certain degree of diversity. In order to speed up the process as a whole, several computers were connected in a client-server concept manner. The server distributes the genotypes from the current cycle population to all available clients, on which side the decoding takes place. Altogether, 15 clients, each equipped with an NVIDIA GTX 1060 graphic card, were available for the NN training process. Therefore, depending on the current population size, complexity of the genomes and number of available clients a cycle took around two to three hours. Overall the elapsed wall time was roughly 120 hours, 87 hours if excluding the two initially completed sessions.

At the end of the final session, the best NN was used for classification. In addition, an ensemble learning strategy was pursued. From a cycle the **10** best individuals could also be selected to vote together on the class of an audio sample. Individuals in higher ranks have more votes to weight them higher. Finally, the class with the most votes wins. In this paper this type of classification is referred to as **population vote**.

4.2. Results

Table 1 illustrates the final results. At the end our best CNN ("DeepSAGA CNN") reached an average accuracy of 72.8 % on the test subset of the development dataset. For the population vote ("Pop. vote") strategy, on the other hand, an average accuracy of 74.7 % was reached.

Scene label	DCASE2018 Baseline	DeepSAGA CNN	Pop. Vote
Airport	72.9 %	84.9 %	85.7 %
Bus	62.9 %	63.2 %	67.4 %
Metro	51.2 %	71.3 %	71.6 %
Metro station	55.4 %	75.3 %	81.9 %
Park	79.1 %	81.0 %	82.2 %
Public square	40.4 %	53.2 %	56.0 %
Shopping mall	49.6 %	75.3 %	73.8 %
Street, pedest.	50.0 %	67.2 %	69.6 %
Street, traffic	80.5 %	85.0 %	86.2 %
Tram	55.1 %	72.0 %	72.4 %
Average	**59.7 %**	**72.8 %**	**74.7 %**

Table 1: The class-wise accuracy for task 1 subtask A evaluated on the test subset of the development dataset.

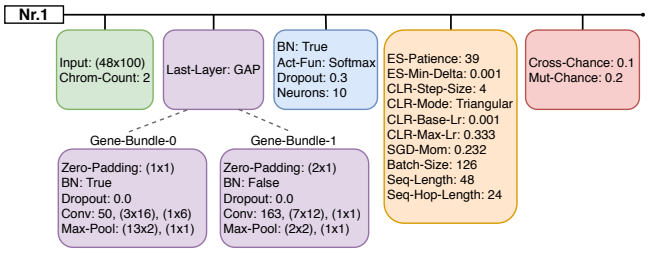


Figure 4: Best genome of the session. GAP: Global-Average-Pooling; BN: Batch-Normalization. The numbers in the brackets are the filter size and the filter stride for the convolutional and max-pooling layers and the first number for the convolutional layer stands for the number of filters.

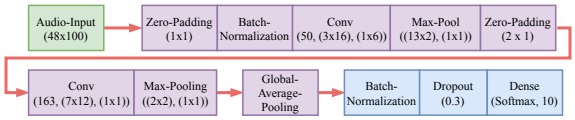


Figure 5: Architecture of "DeepSAGA CNN".

The genome of the "DeepSAGA CNN" can be seen in Figure 4. Figure 5 displays the generated architecture (taking its genome into account) of said CNN. The used hyperparameters can be found in its **Training-Parameters-Chromosome**, which is visible in Figure 4

5. DISCUSSION AND CONCLUSION

In this paper, we described how we developed a genetic algorithm called DeepSAGA to automatically generate CNNs from scratch. Once a session is started, it can be left unattended and offers a selection of NNs after it terminated. We used the DCASE 2018 Challenge as an opportunity to evaluate our algorithm for its competitive ability. With an accuracy of 74.7 % on the test subset the algorithm showed promising results with this specific dataset.

Our investigations throughout an entire session showed that the inclusion of hyperparameters in the search process was an important decision. With regard to their hyperparameters, it often happened that a clone differed (only in its Training-Parameter-Chromosome) from its original only in a few places, but still led to a clear difference in their accuracy. These observations suggest that architectures probably only work well with certain hyperparameter constellations and hyperparameters only with certain architectures.

Throughout the sessions, the approach of population vote resulted in a higher accuracy than that of the best model of the corresponding cycle. A possible reason for that, could be the nature of the population vote itself. Selecting the 10 best CNNs from a cycle, results in predicting with different suitable architectures and hyperparameters simultaneously, while weighting the votes of models with a better fitness higher. Thus, leading to a better overall accuracy as they counterbalance their weaknesses. However, there were fluctuations in the accuracy difference.

In each of our generated CNNs, the "public square" class always proved to have the lowest detection rate. This phenomenon is also reflected in the baseline. A closer look revealed that this class is often mistakenly recognized as a "shopping mall" or "street traffic". In all of these classes background talking is existing and except for the shopping mall traffic noise is involved. Except for adding additional data in future work it could be researched how the evolutionary algorithm could be improved to especially handle the problem with such classes.

Currently DeepSAGA is limited in that way that the architecture after the input layer consists of a series of convolutional layers followed by a series of dense layers. This limitation means that an architecture such as a convolutional layer followed by a dense layer followed by a convolutional layer cannot be created. Additionally, architectures with recurrent layers were not included. Both additions could increase the classification accuracy but also introduce a vast of new parameters that have to be tested by the algorithm if no restrictions are made.

The approach of DeepSAGA is generic and not limitied to audio. Nevertheless, in the future further investigations are required to evaluate its performance with other types of data and data sets so that it can be further optimized to get nearer to the goal to make handcrafted NN architectures obsolete.

6. REFERENCES

[1] A. Mesaros, T. Heittola, and T. Virtanen, "TUT database for acoustic scene classification and sound event detection." IEEE, 2016, pp. 1128–1132. [Online]. Available: http://ieeexplore.ieee.org/document/7760424/

[2] A. Mesaros, T. Heittola, A. Diment, B. Elizalde, A. Shah, E. Vincent, B. Raj, and T. Virtanen, "DCASE 2017 challenge setup: Tasks, datasets and baseline system," in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2017 Workshop (DCASE2017)*, 2017, pp. 85–92.

[3] M. Kim and L. Rigazio, "Deep clustered convolutional kernels," vol. abs/1503.01824, 2015. [Online]. Available: http://arxiv.org/abs/1503.01824

[4] R. Jozefowicz, W. Zaremba, and I. Sutskever, "An empirical exploration of recurrent network architectures," in *International Conference on Machine Learning*, 2015, pp. 2342–2350.

[5] C. Fernando, D. Banarse, M. Reynolds, F. Besse, D. Pfau, M. Jaderberg, M. Lanctot, and D. Wierstra, "Convolution by evolution: Differentiable pattern producing networks," in *Proceedings of the Genetic and Evolutionary Computation Conference 2016*, ser. GECCO '16. ACM, 2016, pp. 109–116. [Online]. Available: http://doi.acm.org/10.1145/2908812.2908890

[6] G. Morse and K. O. Stanley, "Simple evolutionary optimization can rival stochastic gradient descent in neural networks," in *Proceedings of the Genetic and Evolutionary Computation Conference 2016*, ser. GECCO '16. ACM, 2016, pp. 477–484. [Online]. Available: http://doi.acm.org/10.1145/2908812.2908916

[7] G. F. Miller, "Designing neural networks using genetic algorithms." 1989.

[8] K. O. Stanley and R. Miikkulainen, "Evolving neural networks through augmenting topologies," vol. 10, no. 2, pp. 99–127, 2002.

[9] C. Kroos and M. Plumbley, "Neuroevolution for sound event detection in real life audio: A pilot study," in *DCASE 2017*, T. Virtanen, A. Mesaros, T. Heittola, A. Diment, E. Vincent, E. Benetos, and B. Elizalde, Eds. Tampere University of Technology, 2017. [Online]. Available: http://epubs.surrey.ac.uk/842496/

[10] E. Real, S. Moore, A. Selle, S. Saxena, Y. L. Suematsu, J. Tan, Q. Le, and A. Kurakin, "Large-scale evolution of image classifiers," 2017. [Online]. Available: https://arxiv.org/abs/1703.01041

[11] A. Martn, R. Lara-Cabrera, F. Fuentes-Hurtado, V. Naranjo, and D. Camacho, "EvoDeep: A new evolutionary approach for automatic deep neural networks parametrisation," vol. 117, pp. 180–191, 2018.

[12] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," vol. 323, p. 533, 1986. [Online]. Available: http://dx.doi.org/10.1038/323533a0

[13] A. Eiben and J. Smith, *Introduction to Evolutionary Computing*, ser. Natural Computing Series. Springer Berlin Heidelberg, 2015. [Online]. Available: http://link.springer.com/10.1007/978-3-662-44874-8

[14] T. Bäck and A. E. Eiben, "An emperical study on GAs without parameters," in *International Conference on Parallel Problem Solving from Nature*. Springer, 2000, pp. 315–324. [Online]. Available: https://link.springer.com/chapter/10.1007/3-540-45356-3_31

[15] T. Bäck, "The interaction of mutation rate, selection, and self-adaptation within a genetic algorithm." in *PPSN*, 1992, pp. 87–96.

[16] A. Mesaros, T. Heittola, and T. Virtanen, "A multi-device dataset for urban acoustic scene classification," 2018. [Online]. Available: http://arxiv.org/abs/1807.09840

[17] B. McFee, C. Raffel, D. Liang, D. P. Ellis, M. McVicar, E. Battenberg, and O. Nieto, "librosa: Audio and music signal analysis in python," in *Proceedings of the 14th python in science conference*, 2015, pp. 18–25. [Online]. Available: http://www.academia.edu/download/40296500/librosa.pdf

DOMAIN TUNING METHODS FOR BIRD AUDIO DETECTION

Sidrah Liaqat, Narjes Bozorg, Neenu Jose, Patrick Conrey, Antony Tamasi and Michael T. Johnson

University of Kentucky Speech and Signal Processing Lab
Electrical Engineering Dept.
Lexington, KY 40506
mike.johnson@uky.edu

ABSTRACT

This paper presents several feature extraction and normalization methods implemented for the DCASE 2018 Bird Audio Detection challenge, a binary audio classification task, to identify whether a ten second audio segment from a specified dataset contains one or more bird vocalizations. Our baseline system is adapted from the Convolutional Neural Network system of last year's challenge winner *bulbul* [1]. We introduce one feature modification, an increase in temporal resolution of the Mel-spectrogram feature matrix, tailored to the fast-changing temporal structure of many song-bird vocalizations. Additionally, we introduce two feature normalization approaches, a front-end signal enhancement method to reduce differences in dataset noise characteristics and an explicit domain adaptation method based on covariance normalization. Results show that none of these approaches gave significant benefit individually, but that combining the methods lead to overall improvement. Despite the modest improvement, this system won the award for "Highest-scoring open-source/reproducible method" for this task.

Index Terms— audio classification, convolutional neural network, bioacoustic vocalization analysis, domain adaptation

1. INTRODUCTION

The DCASE 2018 Bird Audio Detection Challenge (BADC, DCASE 2018 Challenge Task 3) [2] is a binary audio classification task to determine whether a fixed-length ten second audio segment contains one or more bird vocalizations across a wide variety of bird species and background noise environments. This focuses on the challenging problem of domain adaptation, with the evaluation audio segments to be classified identified as coming from one of three different evaluation datasets, one of which is represented in the training data and two of which are not. This problem of dataset adaptation, also referred to as domain adaptation, domain shift, domain transfer, or dataset bias, is of great interest in a number of domains such as image and audio classification. Recently, the success of deep-learning based approaches requiring large amounts of training data have led to an interest in how to adapt existing well-trained models to new, smaller datasets.

The particular domain of the BADC is that of bioacoustics signal processing and analysis. Currently bioacoustics research often requires extensive amounts of manual labor for segmentation, detection and labeling of voice activity from hours of field recordings [3], and because of this automated analysis of bioacoustics data can be a powerful noninvasive and economical tool for monitoring the diversity, migration patterns [4] and ecosystem health [5] of vocally active animal species. In recent years, speech processing and machine learning techniques for human speech have begun to be used to study animal communication for detection and classification, with applications to censusing [6], understanding the effect of noise on animal communication [7], and other areas of acoustic ecology and ethology. The emphasis of the BADC is to develop a highly generalizable and robust bird classification task that is robust across species and acoustic environments. Although it is presented as a detection problem, it is not "detection" in the sense of typical bioacoustics terminology because it does not involve locating the start and end points of the individual vocalizations.

Our team's submission for the BADC is based on a Convolutional Neural Network (CNN) structure adapted from the baseline architecture of last year's challenge *bulbul* [1]. Nearly all of the top performers of last year's challenge were based on a similar structure, using time-frequency features such as Mel-frequency spectrogram or cepstral features as input to a CNN architecture. Using this baseline, we have introduced three specific modifications to the front-end feature processing methods.

The first of these is adjustment of the time and frequency resolution, which was fairly consistent across many of last year's challenge submissions. The idea behind this change, described in more detail in Section 4.1, is that the variations in the vocalizations of many bird species, especially passerines (songbirds), have a much finer spectral and/or temporal structure than human speech.

The second modification, described in Section 4.2, is the introduction of acoustic signal enhancement, specifically Log-Spectral Amplitude (LSA) estimation combined with Iterative Minimal Controlled Recursive Averaging (IMCRA). The idea behind this is not simply for signal enhancement, which does not typically give improvement

to neural-network based speech or audio classification systems, but instead as a type of dataset normalization intended to decrease the differences between the background noise characteristics of the different datasets.

The third modification, described in Section 4.3, is an explicit domain adaptation technique that applies a source-target covariance transform to the underlying features for an input vocalization based on which dataset it is from.

This paper is organized as follows: in the next section a brief description of data used for training and testing the neural network is provided. Section 3 gives an overview of the baseline system, and section 4 introduces each of the improvements that were implemented to the baseline system in further detail. Section 5 gives results and discussion followed by conclusion in Section 6.

2. DATA

The data provided for the challenge consists of audio recordings from three development datasets and three evaluation datasets which are normalized in amplitude, saved as a 16-bit single channel PCM at a 44.1kHz sampling frequency [2]. Each development dataset has a metadata file associated with it, with a binary label to mark bird presence or absence. The labels are manually annotated by visual analysis of the spectrograms and listening to the audio clips, resulting in a small number of mislabeled files.

The development datasets include Birdvox-DCASE-20k, Warblr10k and Freefield1010. The Birdvox-DCASE-20k dataset was recorded during autumn 2015 in Ithaca, NY, USA as part of a bioacoustics monitoring project. About half of the 20000 files contain at least one bird vocalization [8]. The Birdvox-DCASE-20k dataset was originally recorded at a 24kHz sampling rate and was resampled to 44.1kHz to match the other challenge datasets, and therefore contains no content above 12kHz.

The Warblr10k dataset consists of 8000 audio clips recorded using smartphones, crowdsourced by users of Warblr app in the United Kingdom, with 75.6% of the recordings labeled for bird presence. The Freefield1010 dataset [9] consists of 7690 audio segments derived from files with the field-recording tag in the Freesound crowdsourced global audio archive, with about 25% of dataset labeled as having one or more birds present.

The evaluation data includes 2000 files from Warblr10k, 6620 files from Chernobyl, and 4000 files from PolandNFC. The Chernobyl dataset was collected from the Chernobyl Exclusion Zone as a part of the Transfer Exposure Effects (TREE) project to study the long-term effects of the Chernobyl accident on ecology. The PolandNFC dataset was collected along the Baltic coast of Poland during autumn of 2016. In addition, there was a randomly selected smaller subset of the Warblr10k and Chernobyl (but not PolandNFC) evaluation datasets consisting of approximately 1000 files used for posting ongoing results on the challenge leaderboard. We refer to this as the Leaderboard Evaluation dataset and all testing results in this paper are on this dataset unless otherwise specified. Results are given as Area under the Curve (AUC) of the Receiver Operating Characteristic curve of the submitted prediction probabilities.

Each dataset has unique characteristics in terms of ambient background noise, species present in the recordings, and variety of non-avian interfering sound sources. Warblr, the only dataset present in both training and evaluation sets, and Freefield are crowd-sourced and therefore represent a much wider range of background sound sources, but both the datasets are from UK and therefore have similar species. All three other datasets are remote monitoring data with internal consistency across species and conditions, but vary widely in location and habitat.

It should also be noted that all five of these datasets contain a large number and wide range of bird species that includes both passerine and non-passerine species. Passerine vocalizations tend to have distinct song-like patterns moving around a single or dual frequency (due to the dual-frequency action of the syrinx sound production mechanism), while non-passerine vocalizations are often broadband with unique spectral characteristics. The binary task of classifying whether one or more bird calls is present or non-present inherently represents recognition and classification of many different sound event characteristics.

3. BASELINE SYSTEM

The baseline CNN system was modeled after the baseline architecture of last year's challenge *bulbul* [1] architecture as distributed by the challenge organizers. This is a feed forward network with four 2D CNN layers followed by three dense layers, as shown in detail in Figure 1. The neural network was trained using the log Mel filter bank energy features extracted from small frames of each audio signal. Vocalizations are resampled to a 22.05kHz sampling frequency and divided into 46ms frames using a Hamming window function, with a step size of 14ms, yielding an overlap of 70% across frames. A Fast Fourier Transform is computed, and then Mel filter banks with 80 bands are calculated across a frequency range from 50Hz to 12kHz. The logarithm of the normalized sum magnitude of the filter bank energies is computed for each window. These features were normalized to range between 0 and 1 before feeding to the network input.

Batch normalization layers along with dropout layers were employed in the neural network for improved regularization. The dropout layer has a dropout rate of 0.5. The network also uses L2 layer at the end of CNN layers with a regularization parameter of 0.01. For training, ADAM optimizer is used with an initial learning rate of 0.001. The learning rate was reduced by a factor of 0.2 if there was no improvement in validation accuracy over five consecutive

epochs. The network is trained on binary cross entropy loss using accuracy as a metric. Intermediate activation layers were leaky RELU with a final prediction probability output computed using a sigmoid activation function. Training was done on batches of 16 audio samples over 30 to 40 epochs. Optional data augmentation was built into the system, using a simple cyclic pitch and time shifting approach. For this, the pitch shift was limited to 5% but cyclic time shift could be as much as 90%.

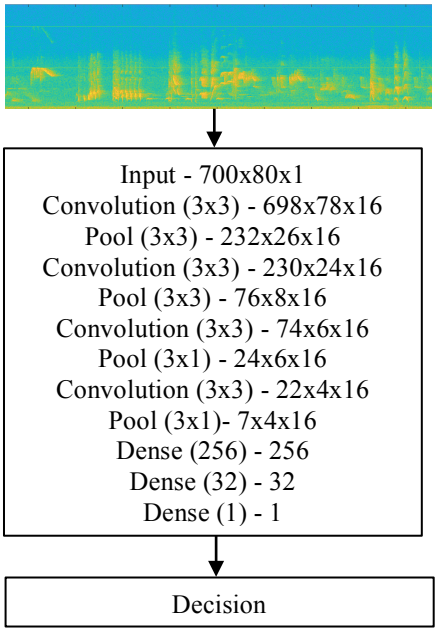


Figure 1: Baseline network architecture

Experiments were conducted both within individual datasets and across datasets using the developmental datasets. Since the number of positive examples varies within each dataset, the selection of positive and negative examples was equalized using class weights to avoid a mismatch in class representations. The dataset organization used for the experiments is given in Table 1 and Table 2.

For testing on the final evaluation dataset, the network was trained on all three development datasets, Birdvox-DCASE-20k, Warblr10k and Freefiled1010, combined. The datasets were shuffled to keep the training of the network impervious to the sequence in which examples are presented to it. Since the number of examples in Freefield1010 and Warblr10k datasets is almost equal, the class imbalance evens out, so equal weights were assigned to positive and negative examples. The development dataset was split into 33905 training examples (95%) and 1784 validation examples (0.05%). The test dataset consisted of the complete evaluation dataset having 12620 examples; 6620 from Chernobyl, 2000 from warblr10k-eval and 4000 from PolandNFC dataset.

Table 1: Within dataset experiments

Dataset	Train (80%)	Test (0.15%)	Validation (.05%)
Birdvox_20k	16000	3000	1000
Freefiled1010	6152	1153	385
Warblr10k	6400	1200	400

Table 2: Cross dataset experiments

Training datasets (84% training and 16% validation)	Test dataset	Class weights	
		-ve	+ve
BirdVox+ freefield	Warblr	43%	57%
Freefield + warblr	BirdVox	50%	50%
BirdVox + Warblr	Freefield	57%	43%

4. PROPOSED IMPROVEMENTS AND RESULTS

4.1. Temporal and frequency resolution

Most bioacoustics signals are nonstationary, like human speech, with changing frequency content over time. Choosing the frame length is a tradeoff between spectral and temporal resolution, with a long frame yielding better spectral resolution but poorer temporal resolution, and vice versa. Bird vocalizations, especially passerine songs, typically have higher frequencies and very fast temporal patterns compared to human speech, with modulations as fast as a few milliseconds [10]. Most of the previous challenge systems, including the baseline bulbul system, have a window size that is relatively long for typical song-bird vocalizations, which would prevent feature representation of small time-scale modulations and transients. Prior work for the BirdClef2017 challenge has also considered resolution issues for bird call recognition [11].

To investigate this, we experimented with changing both the temporal resolution by varying the step and window sizes, as well as changing the frequency resolution by varying the number of filter banks. The high temporal resolution condition used a window size of 12ms with 80 Mel-spaced filter banks (dimension 1669x80), while the high spectral resolution condition used a window size of 32ms along with 160 Mel-spaced filter banks (dimension 624x160). Leaderboard Evaluation results, shown in Table 3 below, indicate that the increased temporal resolution has little impact while the increased spectral resolution has a negative impact.

Table 3: Temporal and spectral resolution results

	AUC	Acc	Val acc
Baseline (B)	86.83	0.89	0.88
High-res temporal (HT)	86.43	0.90	0.89
High-res frequency (HF)	83.54	0.89	0.87

4.2. Signal enhancement

In noisy environments, anthropogenic noise and adverse causes may mask bird song, especially the notes occurring at lower frequencies. In urban environments, birds may modify their songs to low frequency regions to minimize masking effect by anthropogenic noise [12]. Each of the datasets in the BADC has a unique set of background noise characteristics. Our hypothesis for the cross-dataset conditions of the BADC is that applying a front-end signal enhancement may increase similarity across datasets and allow the network to generalize to new noise conditions.

To investigate this, we used the Improved Controlled Recursive Algorithm (IMCRA) noise tracking approach with a log-spectral amplitude estimation technique as proposed by Cohen [13], to implement signal enhancement on all datasets. Noise estimation is updated by averaging the past spectral power values using smoothing parameters that are adjusted with the probability of target signal presence within sub bands. IMCRA includes two iterations of smoothing and minimum tracking. During the first iteration the signal presence probability is detected in each frequency band, and in the second iteration the minimum tracking will be updated by smoothing parameter both in time and frequency domains. This was used with High Temporal features. Results, shown in Table 4 below, show a small degradation to the results from this approach.

Table 4: IMCRA-LSA Signal enhancement results

	AUC	Acc
HT	86.43	0.90
HT enhanced (HTE)	84.47	0.88

4.3. Domain adaptation

One of the primary issues with this challenge problem is the training/test mismatch. There have been a number of different methods suggested for domain adaptation in the image processing literature, to allow well-trained models to be quickly used on new smaller datasets.

In this work we have implemented the CORAL domain adaptation method described in [14] which aligns the second order statistics of source dataset to the target dataset before training the network. This amounts to whitening the training input and then re-coloring it with the covariance characteristic of a chosen target dataset. For this task, we do normalization frame-wise based on the 80 dimensional frequency features. For a baseline 700x80 feature matrix, the normalized feature matrix is

$$\mathbf{A}' = \mathbf{A} \times \left(\mathbf{C}_{\text{source}} + \mathbf{I}\right)^{-\frac{1}{2}} \times \left(\mathbf{C}_{\text{target}} + \mathbf{I}\right)^{\frac{1}{2}}$$

where $\mathbf{A}$ is the original feature matrix, $\mathbf{C}_{\text{source}}$ and $\mathbf{C}_{\text{target}}$ are the 80x80 source and target covariance matrices, and $\mathbf{I}$ is an identity matrix added for regularization.

Although the choice of a target is arbitrary, since the Warblr10k dataset was present in both development and evaluation dataset, we selected it as the target. In the domain adaptation experiments, feature matrices from all other datasets were transformed to the covariance characteristic of the Warblr10k dataset before being applied to the network. Results, shown in Table 5 below, again show little change due to the domain adaptation method.

Table 5: Covariance normalizations results

	AUC	Acc	Val acc
Baseline (B)	86.83	0.89	0.88
Covariance normalized (CN)	86.61	0.87	0.88

4.4. Combined systems

In addition to the individual modifications, several combined systems were implemented. This includes combining high-temporal resolution features with enhancement and covariance normalization, a score fusion system that consisted of a fully connected three-layer neural net using the second from the last layer of each of three individual networks (concatenated 3 32x1 outputs) followed by two dense layers, and several different combinations of simple averaging. Results are shown in Table 6, and indicate that both direct and weighted score fusion methods lead to significant improvement.

Table 6: Composite system results

	AUC
Baseline (B)	86.83
Sequence HT→SE→CV	70.05
Score fusion –Parallel B/HT/HF → 3 FC layers	89.54
Boosting (prediction averaging) (B, HT, HTE, CN, HF)	89.94
Boosting (weighted prediction averaging) (Score Fusion, B, HT, HF)	90.25

5. DISCUSSION AND CONCLUSION

This paper has presented CNN-based methods for the DCASE 2018 Bird Audio Detection challenge, including experiments adjusting the temporal and frequency resolution, signal enhancement for the purpose of dataset normalization, and a method for explicit domain adaptation based on covariance normalizations. Overall results on the leaderboard evaluation dataset show that although none of these approaches gave significant improvements on overall AUC or accuracy metrics, but that combining them together using score fusion approaches were beneficial, improving AUC from a baseline of 86.83 to 90.25 on the leaderboard dataset. The system scored 83.9% on the challenge evaluation dataset, winning the award for Highest-scoring open-source/reproducible method on this challenge task.

6. REFERENCES

[1] Grill, T. and J. Schlüter. *Two convolutional neural networks for bird detection in audio signals*. in *2017 25th European Signal Processing Conference (EUSIPCO)*. 2017.

[2] Stowell, D., et al., *Automatic acoustic detection of birds through deep learning: the first Bird Audio Detection challenge.* 2018.

[3] Christine Erbe, M.L.D., *Animal Bioacoustics.* 2017.

[4] Stepanian, P.M., et al., *Extending bioacoustic monitoring of birds aloft through flight call localization with a three-dimensional microphone array.* Ecol Evol, 2016. **6**(19): p. 7039-7046.

[5] Ross, S.R.P.J., et al., *Listening to ecosystems: data-rich acoustic monitoring through landscape-scale sensor networks.* Ecological Research, 2018. **33**(1): p. 135-147.

[6] Adi, K., M.T. Johnson, and T.S. Osiejuk, *Acoustic censusing using automatic vocalization classification and identity recognition.* J Acoust Soc Am, 2010. **127**(2): p. 874-83.

[7] Catherine, P.O., *Chapter 2: Effects of noise pollution on birds: A brief review of our knowledge.* Ornithological Monographs, 2012. **74**(1): p. 6-22.

[8] Lostanlen, V., Salamon, J., Farnsworth, A., Kelling, S., & Bello, J.P., *Birdvox-full-night : a Dataset and Benchmark for Avian Flight Call Detection.* 2018.

[9] Stowell, D. and M.D. Plumbley, *An open dataset for research on audio field recording archives: freefield1010.* 2013.

[10] Clemins, M.T.J.a.P.J., *Hidden Markov Model Signal Classification*, in *Comparative Bioacoustics: An Overview*. 2017, Bentham Science. p. 358-414.

[11] Sevilla, A. and H. Glotin. *Audio Bird Classification with Inception-v4 extended with Time and Time-Frequency Attention Mechanisms*. in *CLEF*. 2017.

[12] Wood, W.E. and S.M. Yezerinac, *Song Sparrow (Melospiza melodia) Song Varies with Urban Noise (Le Chant de Melospiza melodia Varie avec le Bruit Urbain).* The Auk, 2006. **123**(3): p. 650-659.

[13] Cohen, I., *Noise spectrum estimation in adverse environments: improved minima controlled recursive averaging.* IEEE Transactions on Speech and Audio Processing, 2003. **11**(5): p. 466-475.

[14] Sun, B., J. Feng, and K. Saenko, *Return of Frustratingly Easy Domain Adaptation.* 2015.

ROBUST MEDIAN-PLANE BINAURAL SOUND SOURCE LOCALIZATION

Benjamin R Hammond, Philip J.B. Jackson

Centre for Vision, Speech and Signal Processing, University of Surrey, Guildford, GU2 7XH, UK

ABSTRACT

For a sound source on the median-plane of a binaural system, interaural localization cues are absent. So, for robust binaural localization of sound sources on the median-plane, localization methods need to be designed with this in consideration. We compare four median-plane binaural sound source localization methods. Where appropriate, adjustments to the methods have been made to improve their robustness to real world recording conditions. The methods are tested using different HRTF datasets to generate the test data and training data. Each method uses a different combination of spectral and interaural localization cues, allowing for a comparison of the effect of spectral and interaural cues on median-plane localization. The methods are tested for their robustness to different levels of additive noise and different categories of sound.

Index Terms— Binaural, Localization, Median-Plane

1. INTRODUCTION

Sound source localization using binaural microphones provides an abundance of opportunities for audio augmented reality [1, 2]. Machine based binaural sound source localization could also accompany head mounted display visualizations for the deaf and hard of hearing [3].

Some binaural sound source localization methods assume that the sound source lies on the frontal azimuthal plane [4, 5]. Other methods are designed to localize a sound source on the full-sphere i.e. from any direction of arrival (DOA) around the listener [6, 7]. Some localization methods exploit the movement of the sound source in relation to the listener's head [8, 9]. For non-moving sources, the main localization cues are the interaural cues and spectral cues. In anechoic conditions, for a theoretically symmetrical head, a sound source should produce identical signals at both ears when the direction of arrival of the sound source lies on the median-plane. As such the interaural signal differences will be absent [10, Chapter 2.3]. Although this is the case, some localization methods still implicitly use interaural cues for localization of sound sources that lie on the median-plane [11, 7]. As the interaural signal differences are absent on the median-plane, localization of sound sources on the median-plane using these methods results in high localization errors [6]. As a step towards robust full-sphere binaural sound source localization, there should be a particular focus on the robustness of localization of sound sources that lie on the median-plane.

To generate training data, localization methods make use of a head related transfer function (HRTF) dataset recorded using the same head that recorded the binaural test sound signal. The HRTF describes the frequency based filtering effect of the listener's head, pinna, and torso at the listener's ear canal from a point in space. The time domain equivalent is the head related impulse response (HRIR). A HRTF dataset consists of a collection of measured HRTFs at different DOAs around the listener for both ears [12, Chapter 1]. The spectral cues change between listeners, as the morphology of each listener is different. Therefore, in order to localize a non-moving sound source on the median-plane, the listener's unique HRTF dataset is needed, from which the spectral cues are derived. Methods are rapidly developing for the fast and accurate measurement of personal HRTF datasets in the home environment [13, 14, 15, 16]. When recording HRIRs, unwanted measurement artefacts can be introduced into the recordings. These artefacts are associated with the acoustic environment, the measurement procedure and the post-processing of the data [17, Chapter 8].

Many median-plane localization methods were tested by generating their test data and training data synthetically using the exact same HRTF dataset [18, 19]. This is referred to as the matched condition throughout this paper. With this testing condition, the exact same measurement noise is present in both the test data and training data. For methods tested using this condition, this can actually result in the measurement noise providing an additional localization cue [6]. Methods that are designed and tested only using the matched condition often suffer from overfitting of the training data. To test for robustness, the mismatched condition and off-center condition are also tested. The mismatched condition refers to the case of testing a method using different HRTF datasets to generate the training and test data. For the mismatched condition, the HRTF datasets are captured using the same model of dummy head, but in different rooms, using different measurement equipment. As such the measurement noise is different in the HRTF datasets used to generate the test data and training data, so unlike the matched condition, the measurement noise cannot be negated or used as an additional localization cue. The mismatched condition is a useful testing condition, as for median-plane binaural sound source localization to be possible in real world conditions with the use of a pre-measured HRTF dataset, the method must be robust to additive and convolutive noise provided by the recording equipment. An additional test condition, referred to as the off-center condition uses HRTF templates which have DOAs on the median-plane to localize test sounds generated using HRIR pairs with a lateral angle of 5° to 10° away from the median-plane. This condition is included to test the localization methods' robustness to positioning errors of the loudspeakers and microphones. It is important to test this condition, as even small positioning errors can have a large effect on the interaural cues around the median-plane, especially at higher frequencies.

Reflections from the pinnae create peaks and notches in the spectrum of the sound source as it arrives at the entrance of the ear canal. For different directions of arrival, these peaks and notches occur at different frequencies and have varying degrees of sharpness. The relative level between successive peaks and notches also differs. These peaks and notches predominantly occur in the high-frequency range; approximately above 5kHz [20]. The spectrum

EPSRC Programme Grant S3A: Future Spatial Audio for an Immersive Listener Experience at Home (EP/L000539/1).

of the sound source, as well as additive or convolutive noise provided by the recording equipment may produce confounding peaks and notches in the spectrum of the binaural test sound signal. As such, the localization methods should be tested for their robustness to a variety of different sound sources with different spectral shapes. Ultimately, a localization method should aim to be robust to different recording environments and the presence of reverberation in the binaural recording. However, as the common binaural sound source localization cues are contained within the direct component of the sound, it is first prudent to test the median-plane localization methods in anechoic conditions, i.e. containing only the direct component of the binaural sound signal. As such, the localization methods will be tested using only anechoic conditions in this paper. In other areas of research, accessibility of large scale datasets has made it possible to develop deep learning methods [21]. However there are currently few publicly available HRTF datasets. Each of the methods tested in this paper use only one HRTF dataset to use as templates or generate training data. These methods then have the advantage of performance with a small amount of training data.

2. MEDIAN-PLANE BINAURAL SOUND SOURCE LOCALIZATION

The methods in this paper have been selected to give a diverse representation of the state of the art median-plane sound source localization methods. The first method to be tested is the Peak & Notch Frequency method [22], which estimates the location of a sound source by comparing the estimated frequencies of the peaks and notches in the HRTF of the binaural test sound signal with the estimated frequencies of the peaks and notches in each HRTF of the template HRTF dataset. The peaks and notches are defined as relative maxima and minima in a spectrum respectively. Small spectral fluctuations in the spectrum of the HRTFs and each time frame of the log-magnitude spectrogram of the binaural test sound signal are smoothed using a Gaussian filter. For each time frame of the log-magnitude spectrogram of the binaural test sound signal, a DOA is estimated by a comparison of the frequencies of the peaks and notches in the time frame of the spectrogram to the peaks and notches of each HRTF pair in the training HRTF dataset. The direction of arrival is then estimated as the DOA assigned to the most time frames. For this last step, as there is a large amount of HRTF pairs on the median-plane in the training dataset, it was decided instead to create a PDF as a function of the DOA using a Gaussian kernel smoothing function. The DOA of the sound source is then given by the DOA at the maximum of the PDF. It was found that the frequency range of 4kHz - 18kHz produced the best localization estimates for the mismatched condition.

The second tested method is the Speech Prefilter method [18]. Firstly, a voice activity detector is used to detect voiced speech frames in the binaural sound signal. Time frames that do not have voiced speech detected in them are removed [23]. The prefilter is in the form of cepstral coefficients learned from a training set of speech samples. The received binaural sound signal at the left and right ears are transformed to cepstrograms. Truncated cepstral coefficients are averaged over each time frame of these cepstrograms. The Fourier transform of the summation of the prefilter and the truncated cepstral coefficients yields the estimate of the magnitude of the HRTF for each ear. The estimated magnitude spectrum of the HRTF from the binaural test sound signal for the left and right ears are concatenated, and the magnitude spectrum of the HRTF templates for the left and right ears are also concatenated. Cross-correlation coefficients between these two concatenated spectra are estimated. The DOA of the sound source is estimated by the entry in the database that yields the maximum correlation coefficient. For this paper, the prefilter is trained from approximately one hour of speech sounds from the CSTR VCTK Corpus [24]. The speech samples have been selected to give an equal balance of male and female voices and a diverse set of accents. These training speech samples are not included in the test data. The original method used a narrow frequency range of 3.5kHz - 7.5kHz. For the mismatched condition, it was found that a frequency range of 3.5kHz - 18kHz produced much better localization estimates, as there are more prominent peaks and notches in this range. Additionally, the log-magnitude spectrum is used instead of the magnitude spectrum in all cases.

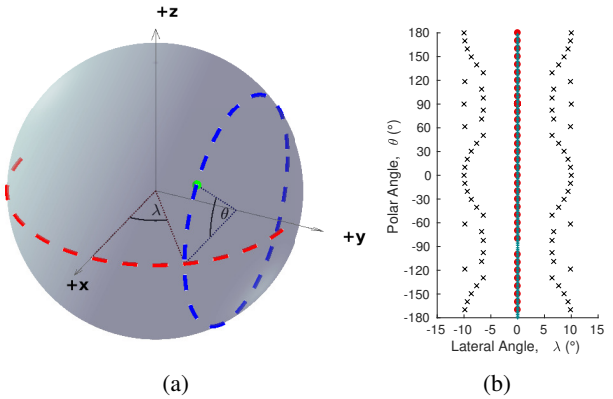
The third tested method is the Cross-Convolution method [25, 19]. For this method, the binaural test sound signal received at the left and right ears are filtered with each contralateral HRIR pair in the training HRTF dataset. Cross-correlation is used to determine the similarity of the left and right ear's filtered observations. The correlation coefficient is calculated using each HRIR pair in the dataset and the maximum correlation coefficient yields the DOA estimate.

The fourth tested method is the MUSIC (MUltiple SIgnal Classification) Signal Subspace method [11]. For this method, the directional information is extracted in narrow subbands from a binaural test sound signal. In order to estimate the location of a sound source, a composite estimator based on signal subspace decomposition is used with a set of HRTF templates from the training HRTF dataset. In order to improve the performance of this method for the mismatched condition, the frequency range is extended to 18kHz. It was additionally found that results improved by using the frequency bins from the STFT of the binaural sound signal instead of filtered narrow subbands. The best results were yielded with an STFT with a window length of 512 samples and a hop size of 128 samples, using a sampling rate of 48kHz.

3. TESTING PROCEDURE

In order to test the robustness of the localization methods, a diverse range of monaural sound sources are used to generate the binaural test sound signals. 10 of these sounds are taken from the environmental sound corpus in [26]. They have been selected for their diversity in spectral characteristics. Namely, they are: Waves crashing, electric saw cutting, water pouring, train moving, chopping wood,typing on keyboard, ice dropping into glass, bells chiming, cars honking and sheep baaing. Additionally, as speech is one of the most important everyday sounds, it is tested under its own category. As such, 10 speech samples have been chosen from the CSTR VCTK Corpus [24]. The speech samples have been selected to give an equal balance of male and female voices and a diverse set of accents. White noise and pink noise are additionally used for testing, giving a total of 22 monaural sound sources used in this paper.

The interaural-polar coordinate system describes the direction of arrival of a sound source with the lateral angle, $\lambda \in [-90^\circ, 90^\circ]$, and the polar angle $\theta \in [-180^\circ, 180^\circ)$ [27], as shown in Figure 1 (a). The HRTF dataset measured in [28] contains 178 HRIR pairs that have DOAs which are approximately evenly spaced on the median-plane. This dataset is referred to as the TH Koln dataset throughout this paper. These 178 HRIR pairs are used for training data by all of the tested methods to estimate the DOA of the sound source. The HRTF dataset used to generate the binaural test sound



(a) (b)

Figure 1: (a) The interaural-polar coordinate system. For a listener at the origin, +x extends directly ahead of the listener with a lateral and polar angle of: $\lambda = 0°$, $\theta = 0°$. The lateral angle range is displayed by the dashed red line. The polar angle range is displayed by the dashed blue line. (b) DOA of HRTFs used for training data and to generate test data. Blue plus sign: DOA of HRTFs used for training data. Red circle: DOA of HRTFs used for the mismatched condition. Black cross: DOA of HRTFs used for the off-center condition.

signals for the mismatched condition and the off-center condition is the dataset measured at RIEC, Tohoku University as part of the "Club Fritz" project [29]. The binaural test sound signals are created synthetically by convolving the HRIR pairs at each of the test positions with each of the 22 monaural sound sources. Stereo uncorrelated pink noise is added to the binaural sound signals to give signal-to-noise ratios (SNRs) from 0dB to 30dB in 10dB steps. For the mismatched condition, 35 HRIR pairs are used, all of which lie on the median-plane and are spaced in 10 degree increments, with the exception of a HRIR pair at $\theta = -90°$, which is absent. For the matched condition, the binaural test sound signals are generated synthetically using the HRIR pairs in the TH Koln dataset with a lateral angle of $\lambda = 0°$ and the nearest polar angle to the DOAs used for the mismatched condition. For the off-center condition, the DOAs of the HRIR pairs used to generate the test data have a lateral angle between and including $\lambda = -10°$ and $\lambda = -5°$, and also between and including $\lambda = 5°$ and $\lambda = 10°$. Within these two lateral angle ranges, the HRIR pairs with a polar angle nearest to the polar angle of the test positions in the mismatched condition are used, giving a total of 70 HRIR pairs used for the off-center condition, as shown in Figure 1 (b).

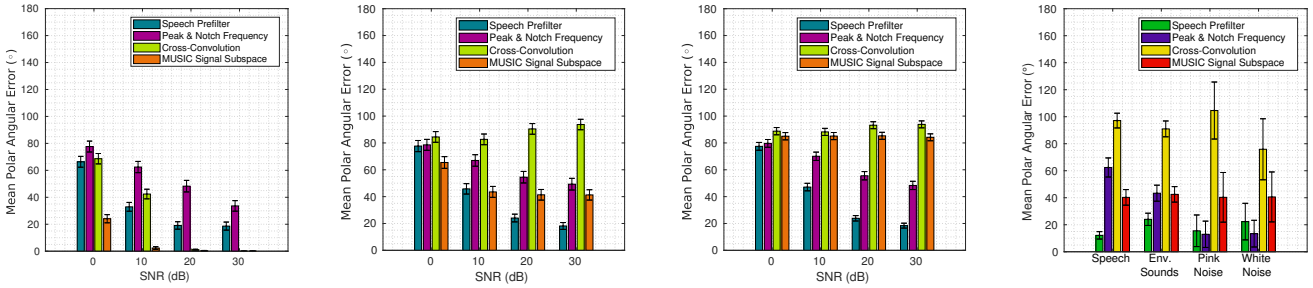
4. RESULTS AND DISCUSSION

Figure 2 (a-c) shows the mean polar angular error for localization on the median-plane. For each testing condition, the polar angular error is the polar angle between the ground truth test position and the estimated position of the sound source, from the point of view of the listener. The results are shown for each of the methods and each of the test conditions. The Speech Prefilter method performs at a similar level for the different testing conditions (matched, mismatched, off-center), though it performs slightly better in the matched condition. Predictably it performs worse with decreasing SNR. The Peak & Notch Frequency method also performs similarly for the different testing conditions. As these two methods only use spectral cues for localization, the interaural cues do not provide any additional benefit for the matched condition, nor do they provide confounding information for the mismatched or off-center conditions. Overall the Speech Prefilter method is better at localization than the Peak and Notch Frequency method. It could be the case that it performs better as it additionally implicitly considers the sharpness and relative level of the peaks and notches, as well as the frequency at which the peaks and notches occur.

For a symmetrical head, the HRTF magnitude spectrum for the right ear should be identical to that of the left ear and the interaural cues should be zero throughout the frequency range for all locations on the median-plane. However a slight error in positioning of the dummy head relative to the loudspeakers results in the peaks and notches for one ear occurring at slightly different frequencies to the those of the other ear. This positioning error results in non-zero values for the interaural cues, with the largest values occurring around the frequencies of the peaks and notches. For the matched condition, this exact same measurement noise exists in the interaural cues for both the binaural test sound signal and HRTF templates, and as such, for this testing condition, the methods that use interaural cues actually exploit this measurement noise as a localization cue. Therefore both the Cross-Convolution method and the MUSIC Signal Subspace method perform very well in the matched condition. However, the exact same measurement noise due to positioning error would not exist in both the binaural test sound signal and HRTF templates in a real world setting and so this demonstrates why developing a median-plane binaural sound source localization method for use with the matched HRTF condition should be avoided. For the mismatched and off-center conditions, the Cross-Convolution method performs very poorly, as the measurement noise in the interaural cues is now different for the binaural test sound signal and HRTF templates. The MUSIC Signal Subspace method implicitly uses both spectral cues and interaural cues. The interaural cues result in the method performing well in the matched condition, however they hinder the method in the mismatched condition and off-center condition.

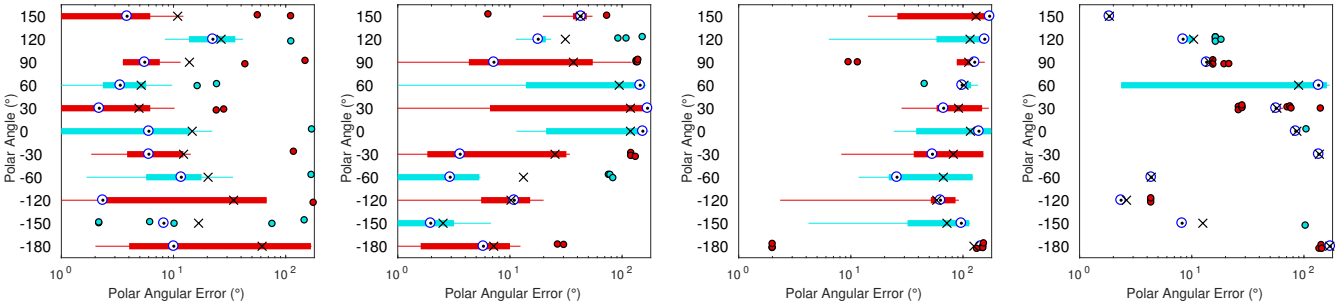
Figure 2 (d) shows the mean polar angular error, as a function of sound category for localization on the median-plane. As the Speech Prefilter method attempts to adjust the spectrum with respect to the average speech spectrum, it performs the best with speech sounds. Pink noise having a spectrum closely resembling speech is the next best at performance with this method, followed by white noise, and then environmental sounds, which have the most confounding spectral cues. The Peak & Notch Frequency method performs best with pink noise and white noise, as they have relatively flat spectra and as such are less prone to producing confounding peaks and notches in the spectrum of the binaural test sound signal. The MUSIC Signal Subspace method performs at a similar level with all sounds, and the cross-convolution method performs poorly in all conditions.

Figure 3 shows the box plot and mean for the polar angular error as a function of the polar angle, for localization on the median-plane. For each polar angle, the results are shown for binaural test sound signals synthetically generated at 30dB SNR with the mismatched condition. The Speech Prefilter method and the Peak & Notch Frequency method perform poorly at 120°, where the HRTF spectrum is relatively flat and has no discernable major peaks or notches. The Speech Prefilter method is also prone to front-back reversals for sound sources at the back of the head, where the first major notch occurs at a similar frequency to the first major notch in the HRTF spectra at the front of the head. The Peak & Notch Frequency method performs best for sound sources underneath the head, where



(a) (b) (c) (d)

Figure 2: Mean polar angular error (°) for localization on the median-plane, as a function of: (a-c) SNR, (d) sound category. The results are shown for binaural test sound signals synthetically generated using: (a-c) All of the monaural sound sources, (d) the monaural sound sources specified by the sound category in the abscissa. The methods shown are the Speech Prefilter, Peak & Notch Frequency, Cross-Convolution, and MUSIC Signal Subspace. The results are shown for: (a) matched condition, (b,d) mismatched condition, (c) off-center condition. The error bars correspond to 95% confidence intervals.



(a) (b) (c) (d)

Figure 3: Polar angular error (°), as a function of polar angle (°) for localization on the median-plane. The results are shown for binaural test sound signals synthetically generated at 30dB SNR. Results are shown for the mismatched condition. The methods shown are: (a) Speech Prefilter, (b) Peak & Notch Frequency, (c) Cross-Convolution, (d) MUSIC Signal Subspace. Cross: Mean; Black dot in white circle: Median; box: Inter-quartile range (IQR); whisker: Within quartile $\pm$ 1.5.IQR; outliers are shown as filled circles.

there are more peaks and notches in the spectrum caused by the HRTF. For the MUSIC Signal Subspace method, the variance is low, indicating that the algorithm is robust to the sound type. Additionally, in Figure 2 (a-c), the performance of the method is fairly consistent for SNRs of 10dB or higher. This would indicate that the method is also robust to additive noise. However, the method suffers in the mismatched condition due to its inherent utilization of interaural cues. The method performs well at angles where the interaural cues of the HRTFs used to generate the binaural test sound signal are similar to the interaural cues of the HRTF templates with the same DOA. However at angles where the interaural cues of the binaural test sound signal and the HRTF template with the same DOA are different, the interaural cues act as confounding cues and the DOA cannot be estimated well.

5. CONCLUSION

In this paper we compared four localization methods for their ability to localize a sound source on the median-plane. Appropriate adjustments have been made to the methods to make them robust to real world recording conditions. The Spectral Prefilter method uses the frequency, sharpness and relative levels of the peaks and notches in the spectrum for localization and it has outperformed the Peak & Notch Frequency method, which only uses the estimated frequencies of the peaks and notches for localization. For methods that use interaural cues for localization, there is a large disparity between the results for the matched and mismatched conditions. This is because the binaural test sound signal and the HRTF template with the same DOA contain the same measurement noise for the matched condition, resulting in the measurement noise being exploited as a localization cue. However, the binaural test sound signal and its corresponding HRTF template would not contain the exact same measurement noise in a real world setting, and so we make the case that future median-plane binaural sound source localization methods should not be designed for use with the matched condition. Furthermore, median-plane binaural sound source localization methods should use spectral cues only, and should not use interaural cues for localization. Future work should consider robustness of the median-plane binaural sound source localization methods to the presence of reverberation.

6. REFERENCES

[1] A. Harma, J. Jakka, M. Tikander, M. Karjalainen, T. Lokki, J. Hiipakka, and G. Lorho, "Augmented reality audio for mobile and wearable appliances," *J. Audio Eng. Soc*, vol. 52, no. 6, pp. 618–639, 2004.

[2] P. F. Hoffmann, F. Christensen, and D. Hammershi, "Insert earphone calibration for hear-through options," in *Audio Engineering Society Conference: 51st International Conference: Loudspeakers and Headphones*, Aug 2013.

[3] D. Jain, L. Findlater, J. Gilkeson, B. Holland, R. Duraiswami, D. Zotkin, C. Vogler, and J. E. Froehlich, "Head-mounted display visualizations to support sound awareness for the deaf and hard of hearing," in *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, ser. CHI '15. New York, NY, USA: ACM, 2015, pp. 241–250.

[4] M. Dietz, S. D. Ewert, and V. Hohmann, "Auditory model based direction estimation of concurrent speakers from binaural signals," *Speech Communication*, vol. 53, no. 5, pp. 592–605, 2011.

[5] J. Woodruff and D. Wang, "Binaural detection, localization, and segregation in reverberant environments based on joint pitch and azimuth cues," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 21, no. 4, pp. 806–815, 2013.

[6] B. Hammond and P. Jackson, "Robust full-sphere binaural sound source localization," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2018)*. IEEE, 2018.

[7] M. Rothbucher, M. Durkovic, T. Habigt, H. Shen, and K. Diepold, "HRTF-based localization and separation of multiple sound sources," in *2012 IEEE RO-MAN: The 21st IEEE International Symposium on Robot and Human Interactive Communication*. IEEE, 2012, pp. 1092–1096.

[8] N. Ma, T. May, H. Wierstorf, and G. J. Brown, "A machine-hearing system exploiting head movements for binaural sound localisation in reverberant conditions," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2015, pp. 2699–2703.

[9] C. Zhou, R. Hu, W. Tu, X. Wang, and L. Gao, "Binaural moving sound source localization by joint estimation of ITD and ILD," in *Audio Engineering Society Convention 130*. Audio Engineering Society, 2011.

[10] J. Blauert, *Spatial Hearing: the psychophysics of human sound localization*, revised ed. Cambridge, MA: MIT press, Cambridge, MA, 1997.

[11] D. S. Talagala, W. Zhang, T. D. Abhayapala, and A. Kamineni, "Binaural sound source localization using the frequency diversity of the head-related transfer function," *The Journal of the Acoustical Society of America*, vol. 135, no. 3, pp. 1207–1217, 2014.

[12] B. Xie, *Head-related transfer function and virtual auditory display*, 2nd ed. Florida, USA: J. Ross Publishing, Inc., 2013.

[13] F. Rumsey, "Spatial audio: Binaural challenges," *J. Audio Eng. Soc*, vol. 62, no. 11, pp. 798–802, 2014.

[14] K. Sunder, J. He, E. L. Tan, and W.-S. Gan, "Natural sound rendering for headphones: Integration of signal processing techniques," *Signal Processing Magazine, IEEE*, vol. 32, no. 2, pp. 100–113, March 2015.

[15] N. D. Hai, N. K. Chaudhary, S. Peksi, R. Ranjan, J. He, and W. S. Gan, "Fast HRFT measurement system with unconstrained head movements for 3d audio in virtual and augmented reality applications," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, March 2017, pp. 6576–6577.

[16] R. Ranjan, J. He, and W.-S. Gan, "Fast continuous acquisition of HRTF for human subjects with unconstrained random head movements in azimuth and elevation," in *Audio Engineering Society Conference: 2016 AES International Conference on Headphone Technology*, Aug 2016.

[17] H. Kuttruff, *Room acoustics*, 5th ed. Abingdon, UK: Spon Press, 2009.

[18] D. S. Talagala, X. Wu, W. Zhang, and T. D. Abhayapala, "Binaural localization of speech sources in the median plane using cepstral HRTF extraction," in *2014 22nd European Signal Processing Conference (EUSIPCO)*, Sept 2014, pp. 2055–2059.

[19] M. Usman, F. Keyrouz, and K. Diepold, "Real time humanoid sound source localization and tracking in a highly reverberant environment," in *Signal Processing, 2008. ICSP 2008. 9th International Conference on*. IEEE, 2008, pp. 2661–2664.

[20] K. Iida, M. Itoh, A. Itagaki, and M. Morimoto, "Median plane localization using a parametric model of the head-related transfer function based on spectral cues," *Applied Acoustics*, vol. 68, no. 8, pp. 835 – 850, 2007.

[21] C. Rascon and I. Meza, "Localization of sound sources in robotics: A review," *Robotics and Autonomous Systems*, vol. 96, pp. 184 – 210, 2017.

[22] E. Blanco-Martin, F. J. Casajus-Quiros, J. J. Gomez-Alfageme, and L. I. Ortiz-Berenguer, "Estimation of the direction of auditory events in the median plane," *Applied Acoustics*, vol. 71, no. 12, pp. 1211–1216, 2010.

[23] Z.-H. Tan and B. Lindberg, "Low-complexity variable frame rate analysis for speech recognition and voice activity detection," *IEEE Journal of Selected Topics in Signal Processing*, vol. 4, no. 5, pp. 798–807, 2010.

[24] C. Veaux, J. Yamagishi, K. MacDonald, *et al.*, "CSTR VCTK Corpus: English Multi-speaker Corpus for CSTR Voice Cloning Toolkit," 2017.

[25] M. Rothbucher, D. Kronmüller, K. Diepold, M. Durkovic, and T. Habigt, *HRTF sound localization*. INTECH Open Access Publisher, 2011.

[26] B. Gygi, G. R. Kidd, and C. S. Watson, "Similarity and categorization of environmental sounds," *Perception & psychophysics*, vol. 69, no. 6, pp. 839–855, 2007.

[27] R. Baumgartner, P. Majdak, and B. Laback, "Assessment of sagittal-plane sound localization performance in spatial-audio applications," in *The technology of binaural listening*. Springer, 2013, pp. 93–119.

[28] B. Bernschütz, "A spherical far field HRIR/HRTF compilation of the Neumann KU 100," in *Proceedings of the 40th Italian (AIA) Annual Conference on Acoustics and the 39th German Annual Conference on Acoustics (DAGA) Conference on Acoustics*, 2013, p. 29.

[29] A. Andreopoulou, D. R. Begault, and B. F. Katz, "Inter-laboratory round robin HRTF measurement comparison," *IEEE Journal of Selected Topics in Signal Processing*, vol. 9, no. 5, pp. 895–906, 2015.

ITERATIVE KNOWLEDGE DISTILLATION IN R-CNNS FOR WEAKLY-LABELED SEMI-SUPERVISED SOUND EVENT DETECTION

Khaled Koutini, Hamid Eghbal-zadeh, Gerhard Widmer

Institute of Computational Perception (CP-JKU),
Johannes Kepler University Linz, Austria
khaled.koutini@jku.at

ABSTRACT

In this paper, we present our approach used for the CP-JKU submission in Task 4 of the DCASE-2018 Challenge. We propose a novel iterative knowledge distillation technique for weakly-labeled semi-supervised event detection using neural networks, specifically Recurrent Convolutional Neural Networks (R-CNNs). R-CNNs are used to tag the unlabeled data and predict strong labels. Further, we use the R-CNN strong pseudo-labels on the training datasets and train new models after applying label-smoothing techniques on the strong pseudo-labels. Our proposed approach significantly improved the performance of the baseline, achieving the event-based f-measure of 40.86% compared to 15.11% event-based f-measure of the baseline in the provided test set from the development dataset.

Index Terms— Weakly-labeled, Semi-supervised, Knowledge Distillation, Recurrent Neural Network, Convolutional Neural Network

1. INTRODUCTION

Motivated by the release of Audioset [1], the task of predicting strong labels using models trained on weakly-labeled audio data was introduced in the DCASE-2017 challenge (task 4) [2]. However, in DCASE-2018, the task has changed and transformed into a semi-supervised task which adds another dimension of complexity to this challenge. By leaving the majority of the training data unlabeled [3], the organizers motivated the participants to leverage the large sets of unlabeled data in a semi-supervised manner in order to improve the performance of their systems. Another important change compared to DCASE-2017 is the evaluation metric, that is changed from segment-based evaluation to event based evaluation. In DCASE-2018 task4, the submissions will be evaluated by the macro average of class-wise *event-based* F1-scores (explained in Section 4.3). The new evaluation metric introduces new challenges to the task, since the systems need to predict the onsets and offsets of the events very accurately. In other word, unlike DCASE-2017, events that are partially detected – with inaccurate onsets and offsets– do not improve the performance based on the new evaluation metric, but rather worsen it, as it will get evaluated as both a false positive and a false negative [3]. In this paper, we propose a novel approach to overcome the difficulties of this new task by leveraging the unlabeled data via an iterative knowledge distillation in neural networks. We show that using our method, the performance of a Convolutional Recurrent Neural Network (R-CNN) can be significanlt improved. We provide experimental results on DCASE-2018 task 4 dataset and compare it with the baselines we used. The remainder of the paper is as follows. Section 2 describes the related work. In Section 3 we explain our proposed method. The experiments and the empirical results are presented in Section 4 and finally Section 5 concludes the paper.

2. RELATED WORK

2.1. Weakly Labels Sound event detection

To deal with weak labels, it is important to pay attention to the power of state-of-the-art tagging systems. By using a R-CNN architecture, Xu et al. [4] achieved the best tagging performance in DCASE 2017 task4. Their architecture uses gated activations of convolutional and recurrent layers and an attention mechanism to locate the events. Their architecture consists of multiple gated convolutional layers followed by a bi-directional Gated Recurrent Unit (GRU). Between convolutional blocks, they used max-pooling only on the frequency dimension, in order to keep the time information required for event localization.

Lee et al. [5] used an ensemble of multiple deep convolution neural networks trained on audio clips of different lengths and managed to achieve the best event detection performance in DCASE 2017 task 4. They showed the power of an ensemble model for such tasks, following an ensemble method proposed by Caruana et al. [6] by iteratively adding models that increase the performance of the whole system.

2.2. Knowledge Distillation In Neural Networks

A considerable amount of work has been done in transferring the knowledge between models either for compressing models while maintaining their performance [7, 8, 9, 10] or for increasing the interpretability and explaining the decisions [11, 12, 13]. A pioneer idea of knowledge transfer from a large model or an ensemble of multiple models to a simple model was introduced by Bucila et al. [7] in the context of compressing large models into small models that are more suitable for deployment. Ba and Caruana [8] empirically showed that a similar performance to the state-of-the-art deep neural network models can be achieved using shallow models. This performance of shallow models can not be achieved by training on the original training data, but rather by training shallow model to mimic the output activations of a deep model. Further work by Hinton et al. [9] showed that these simple models (also known as student model) can even perform better than the models they mimic, by distilling the knowledge from an ensemble of deeper models (known as teacher models) into a single new model (the student). They managed to improve the performance of their models, both on the MNIST dataset [14] and for an Automatic Speech Recognition task (ASR).

Furlanello et al. [10] managed to make student models surprisingly outperform their teacher models on many computer vision and language modeling tasks, by retraining the student models with identical parameterization to their teachers, but with different initialization. They trained the student models to predict the correct labels, and further to match the output distribution of the teacher.

3. THE PROPOSED APPROACH

In this section, we detail the key components of our proposed iterative knowledge distillation method.

3.1. Key Differences With Previous Work

We adopted a deep architecture (described in Table 2) inspired by the one proposed by Xu et al. [4]. However, We used the ReLu activation function for the convolutional layers and kept the gated linear activation only for the recurrent layer. we used the Simple Recurrent Unit (SRU) [15] as a recurrent unit because of its fast training. We achieved empirically a better tagging performance using a global average of the frame level probabilities, instead of the attention mechanism proposed by Xu et al.. Our shallow model (Table 3) is inspired by the DCASE-2018 task 4 baseline model [3] with an adjusment of replacing the recurrent unit in the baseline system with an SRU.

Unlike the approaches stated in Section 2.2, we trained our new student models on the smoothed predictions over the time-dimension of the teacher models. We show that smoothing is an important step given the nature of our task. Namely, we used *median smoothing* with varying window size and with/without Gaussian filter smoothing (Figure 1). We repeated this step a couple of times, although the improvement was diminishing over the steps. We also trained deep and shallow models in each iteration, as follows. We used the predictions of the best model for each class as the pseudo-labels of the next iteration for knowledge distillation. In comparison with Furlanello et al. [10], they trained the new models with the supervision of only the latest iteration of a single model, while Hinton [9], Bucila [7] and their collaborators used an ensemble of teacher models in a non-iterative manner. And finally, we used the smoothed labels, while the aforementioned methods use the probabilities of the teacher to train the students.

3.2. Proposed Approach for Audio Tagging

We train an R-CNN on the weakly-labeled dataset and predicted pseudo-weak-labels for both in-domain and out-of-domain sets. Table 2 shows the configuration of the layers of the model.

3.3. The Proposed Approach for Strong Label Prediction

We follow a multi-pass strategy to get our final predictions, by iteratively predicting pseudo-strong-labels for the labeled, in-domain and out-of-domain sets, and retraining new models on those new predictions.

3.3.1. The First Pass

We trained a recurrent convolutional neural network with the same architecture that was used for tagging (Table 2). However, the network is not only trained on the provided labels of the labeled set, but also on the predicted pseudo labels for both the in-domain and out-of-domain sets. The result of the first pass are strong labels for the labeled, in-domain and out-of-domain sets. These labels are presented in the form of frame-level probabilities for every audio clip.

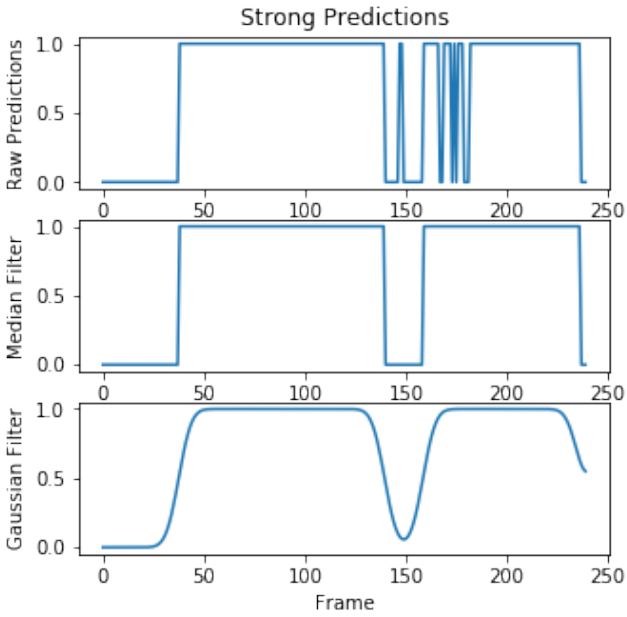


Figure 1: Example of strong predictions before/after smoothing.

3.3.2. The Second Pass

In the second pass, we smooth the current predicted pseudo-strong labels using median/Gaussian filters and we train new models on them. We observed that the performance of the models varies among different classes. We achieved better performances in some classes using a deep model (Table 2), while for other classes shallow models (Table 3) performed better. In addition, using median smoothing with or without Gaussian smoothing resulted in varying performances for different classes.

3.3.3. Model selection

We train multiple models with/without smoothing. Then, we select the best trained model for each class to predict new pseudo-strong-labels for the respected class for the labeled, in-domain and out-of-domain sets. Using these new prediction, we iteratively repeated the second pass (Figure 2).

3.3.4. Smoothing for Strong Prediction

The strong predictions of our models trained only on weakly-labeled data tend to be noisy. Therefore, we smooth those predictions using median and Gaussian filters (Figure 1). We then use these smoothed probabilities for retraining the network in the next pass as explained in Section 3.3.

Table 1: F-score results per class by pass. The average is calculated class-wise (macro-average) [16]. SM: shallow model (Table 3). DM: Deep model (Table 2). Merge: merging models by taking the prediction of the best model for each class. Note that for pass 1 (marked with *) the models are trained using weak-labels. From pass 2 onwards, the models are trained on the smoothed strong predictions of the previous pass.

Pass	Config.	Average	Alarm	Blender	Cat	Dishes	Dog	Electric..	Frying	Runnin..	Speech	Vacuum..
1*	SM	17.43	7.1	12.3	2.2	4.9	4.8	38.7	36.6	13.0	2.5	52.1
1*	DM	24.65	26.0	24.3	26.3	13.2	23.5	33.3	12.4	16.2	41.8	29.4
2	SM	35.18	39.8	33.3	32.1	15.8	25.3	40.0	38.8	22.5	47.3	56.9
2	DM	34.08	33.9	31.9	32.1	15.4	24.9	39.3	37.1	22.5	47.3	56.3
3	SM	34.46	41.8	32.5	33.3	16.1	16.3	40.0	40.0	22.4	47.1	55.1
3	DM	33.59	37.3	33.8	30.0	15.6	20.5	35.7	36.6	20.9	48.3	57.1
4	SM	34.46	41.8	32.5	33.3	16.1	16.3	40.0	40.0	22.4	47.1	55.1
4	DM	34.67	42.5	32.8	32.6	14.0	21.4	42.9	37.7	20.9	46.9	54.9
5	SM	35.48	43.9	38.3	31.1	13.1	21.5	40.6	41.3	22.1	48.2	54.7
5	DM	34.85	40.4	39.5	33.5	14.5	20.6	42.1	39.3	18.1	46.4	54.0
6	Merge	37.89	46.6	38.2	45.6	14.2	24.2	41.3	40.0	28.0	47.6	53.3
7	Merge	39.11	46.0	39.5	41.1	18.1	25.7	43.3	43.1	28.4	48.7	57.1
8	Merge	40.86	49.3	40.0	50.0	18.1	25.7	44.1	43.5	31.0	49.9	57.1

Table 2: Proposed deep architecture for predicting strong labels and audio tagging. BN: Batch normalization, BIAS: Model uses bias with no batch normalization, ReLu: Rectified Linear activation function

Input 240 × 64	
2 × 2 Conv(pad-1, stride-1)-64-BN-ReLu 2 × 2 Conv(pad-1, stride-1)-64-BN-ReLu 1 × 2 Max-Pooling	
2 × 2 Conv(pad-1, stride-1)-64-BN-ReLu 2 × 2 Conv(pad-1, stride-1)-64-BN-ReLu 1 × 2 Max-Pooling	
2 × 2 Conv(pad-1, stride-1)-64-BN-ReLu 2 × 2 Conv(pad-1, stride-1)-64-BN-ReLu 1 × 2 Max-Pooling	
2 × 2 Conv(pad-1, stride-1)-64-BN-ReLu 2 × 2 Conv(pad-1, stride-1)-64-BN-ReLu 1 × 2 Max-Pooling	
1 × 1 Conv(pad-1, stride-1)-256-BIAS-ReLu 1 × 4 Max-Pooling	
Bi-directional SRU 128 hidden units	
1 × 1 Conv(pad-1, stride-1)-10-BIAS-Sigmoid Output 240 × 10	
(Strong predictions) Output 240 × 10	(Weak-label training and tagging) Global-Average-Pooling Output 10

Table 3: Proposed shallow architecture for predicting strong labels. Similar to the baseline [3]. BN: Batch normalization, BIAS: Model uses bias with no batch normalization, ReLu: Rectified Linear activation function

Input 240 × 64	
3 × 3 Conv(pad-1, stride-1)-64-BN-ReLu 1 × 4 Max-Pooling	
3 × 3 Conv(pad-1, stride-1)-64-BN-ReLu 1 × 4 Max-Pooling	
3 × 3 Conv(pad-1, stride-1)-64-BN-ReLu 1 × 4 Max-Pooling	
Bi-directional SRU 128 hidden units	
1 × 1 Conv(pad-1, stride-1)-10-BIAS-Sigmoid Output 240 × 10	
(Strong predictions) Output 240 × 10	(Weak-label training and tagging) Global-Average-Pooling Output 10

4. EXPERIMENTS AND RESULTS

4.1. Dataset

The dataset is split into a training set, a test set and an evaluation set [3]. The training set contains three subsets, a labeled set, an unlabeled-in-domain set and an unlabeled-out-of-domain set. In this paper, they are referred to as labeled, in-domain, out-of-domain respectively. The test set contains 288 strongly labeled audio clips. The evaluation set consist of 880 audio clips, for which our system predicted strong labels for the challenge submission.

4.2. Features Extraction

We use log-scaled Mel-bands spectrograms as an input for all our models. We extracted 64 Mel bands from 64 ms frames with 22.5 ms overlap using Librosa [17]. That resulted in an input size of 240 × 64 for our models.

4.3. Evaluation Metric

The evaluation metric for the task is the event-based F-score [16]. The predicted events are compared with a reference event list, by comparing the onset and the offset of the predicted event with the overlapping reference event. The predicted event is considered correctly detected (true positive), if it's onset is within 200 ms collar of the reference event onset and its offset is within 200 ms or 20% of the event length collar around the reference offset. If a reference event has no matching predicted event, it is considered a false negative. If the predicted event doesn't match any reference event, it is considered a false positive. Furthermore, if the system partially predicted an event without accurately detecting its onset and offset,

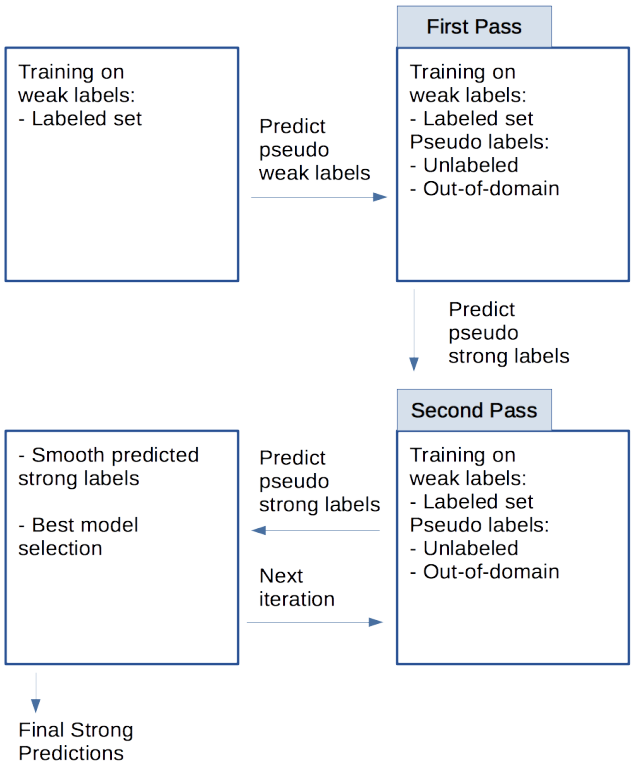


Figure 2: The proposed knowledge distillation framework for RCNNs.

it will be penalized twice, as a false positive and a false negative. Equation (1) shows the calculation of the F-score for each class [3].

$$F_c = \frac{2.TP_c}{2.TP_c + FP_c + FN_c}, \quad (1)$$

Where F_c, TP_c, FP_c, FN_c are the F-score, true positives, false positives, false negatives of the class c respectively. The final evaluation metric the average of the F-score for all the classes.

4.4. Results

Table 1 shows the class-wise intermediate results over the training iterations. In the first pass, the shallow and the deep models where trained on the given weak labels and on the predicted pseudo weak labels for the in-domain and out-of-domain sets. We show that the shallow model works better only for the classes Electric shaver/toothbrush, frying and vacuum cleaner. We justify that by the nature of these classes, as they tend to be longer, with the event-length medians 8.78, 10.00, 9.99 respectively, compared to 1.03 the event-length median for all the classes (Table 1 in [3]). Therefore, we conclude that the shallow model fails to localize when trained on weak labels. However, the shallow models works surprisingly well when trained on the strong prediction of the previous pass. They even generalize better in many cases then the deep models (passes 2 to 5 for many classes). By merging the predictions of the best model for each class iteratively, we managed to push the performance of the system to 40.86%.

Table 4 shows the final macro-averaged event-based evaluation results on the test set compared to the baseline system.

Table 4: The performance of our approach compared to the baseline system [3]. Note that we re-ran the baseline on our machines, hence the slight difference from the reported values in [3].

	F1	Precision	Recall
Baseline	15.11	14.20	17.80
Our system	40.86	40.21	44.42

5. CONCLUSION

In this paper we propose a method for detecting sound events from weakly-labeled data. We proposed iteratively training similar models with different initialization on the smoothed predictions of the previous iteration. The goal behind this is to iteratively making the detected sound events more precise and predicting the onsets and the offsets of the events more accurately. We provide empirical evidence that this iterative process makes the predicted time boundaries for individual events more accurate, in accordance with the results of [10]. The event-based F-score increases over iterations to reach 40.86% on the test set, compared to the baseline performance of 15.11%. We also show empirically that shallow models trained on the predictions of deep models can even generalize better then their teachers, in line with the results of [8].

6. ACKNOWLEDGMENT

This work has been supported by (1) the COMET-K2 Center for Symbiotic Mechatronics of the Linz Center of Mechatronics (LCM) and (2) the COMET Center SCCH, with funds provided by the Austrian Federal Government, the Federal State of Upper Austria, and the Austrian Ministries BMVIT and BMWFW.

7. REFERENCES

[1] J. F. Gemmeke, D. P. Ellis, D. Freedman, A. Jansen, W. Lawrence, R. C. Moore, M. Plakal, and M. Ritter, "Audio set: An ontology and human-labeled dataset for audio events," in *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*. IEEE, 2017, pp. 776–780.

[2] A. Mesaros, T. Heittola, A. Diment, B. Elizalde, A. Shah, E. Vincent, B. Raj, and T. Virtanen, "Dcase 2017 challenge setup: Tasks, datasets and baseline system," in *DCASE 2017-Workshop on Detection and Classification of Acoustic Scenes and Events*, 2017.

[3] R. Serizel, N. Turpault, H. Eghbal-Zadeh, and A. Parag Shah, "Large-Scale Weakly Labeled Semi-Supervised Sound Event Detection in Domestic Environments," July 2018, submitted to DCASE2018 Workshop. [Online]. Available: https://hal.inria.fr/hal-01850270

[4] Y. Xu, Q. Kong, W. Wang, and M. D. Plumbley, "Large-scale weakly supervised audio classification using gated convolutional neural network," *arXiv preprint arXiv:1710.00343*, 2017.

[5] D. Lee, S. Lee, Y. Han, and K. Lee, "Ensemble of convolutional neural networks for weakly-supervised sound event detection using multiple scale input," Tech. Rep., DCASE2017 Challenge, Tech. Rep., 2017.

[6] R. Caruana, A. Niculescu-Mizil, G. Crew, and A. Ksikes, "Ensemble selection from libraries of models," in *Machine Learning, Proceedings of the Twenty-first International Conference (ICML 2004), Banff, Alberta, Canada, July 4-8, 2004*, 2004. [Online]. Available: http://doi.acm.org/10.1145/1015330.1015432

[7] C. Bucila, R. Caruana, and A. Niculescu-Mizil, "Model compression," in *Proceedings of the Twelfth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Philadelphia, PA, USA, August 20-23, 2006*, 2006, pp. 535–541. [Online]. Available: http://doi.acm.org/10.1145/1150402.1150464

[8] J. Ba and R. Caruana, "Do deep nets really need to be deep?" in *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, 2014, pp. 2654–2662. [Online]. Available: http://papers.nips.cc/paper/5484-do-deep-nets-really-need-to-be-deep

[9] G. E. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *CoRR*, vol. abs/1503.02531, 2015. [Online]. Available: http://arxiv.org/abs/1503.02531

[10] T. Furlanello, Z. C. Lipton, M. Tschannen, L. Itti, and A. Anandkumar, "Born-again neural networks," in *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, 2018, pp. 1602–1611. [Online]. Available: http://proceedings.mlr.press/v80/furlanello18a.html

[11] R. Chandra, K. Chaudhary, and A. Kumar, "The combination and comparison of neural networks with decision trees for wine classification."

[12] S. Tan, R. Caruana, G. Hooker, and A. Gordo, "Transparent model distillation," *arXiv preprint arXiv:1801.08640*, 2018.

[13] N. Frosst and G. E. Hinton, "Distilling a neural network into a soft decision tree," in *Proceedings of the First International Workshop on Comprehensibility and Explanation in AI and ML 2017 co-located with 16th International Conference of the Italian Association for Artificial Intelligence (AI*IA 2017), Bari, Italy, November 16th and 17th, 2017.*, 2017. [Online]. Available: http://ceur-ws.org/Vol-2071/CExAIIA_2017_paper_3.pdf

[14] L. Deng, "The mnist database of handwritten digit images for machine learning research [best of the web]," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, Nov 2012.

[15] T. Lei and Y. Zhang, "Training rnns as fast as cnns," *arXiv preprint arXiv:1709.02755*, 2017.

[16] A. Mesaros, T. Heittola, and T. Virtanen, "Metrics for polyphonic sound event detection," *Applied Sciences*, vol. 6, no. 6, p. 162, 2016.

[17] B. McFee, C. Raffel, D. Liang, D. P. Ellis, M. McVicar, E. Battenberg, and O. Nieto, "librosa: Audio and music signal analysis in python," in *Proceedings of the 14th python in science conference*, 2015, pp. 18–25.

TRAINING GENERAL-PURPOSE AUDIO TAGGING NETWORKS WITH NOISY LABELS AND ITERATIVE SELF-VERIFICATION

Matthias Dorfer

Johannes Kepler University
Institute of Computational Perception
Linz, 4040, Austria
matthias.dorfer@jku.at

Gerhard Widmer

Johannes Kepler University
Institute of Computational Perception
Linz, 4040, Austria
gerhard.widmer@jku.at

ABSTRACT

This paper describes our submission to the first Freesound general-purpose audio tagging challenge carried out within the DCASE 2018 challenge. Our proposal is based on a fully convolutional neural network that predicts one out of 41 possible audio class labels when given an audio spectrogram excerpt as an input. What makes this classification dataset and the task in general special, is the fact that only 3,700 of the 9,500 provided training examples are delivered with manually verified ground truth labels. The remaining non-verified observations are expected to contain a substantial amount of label noise (up to 30-35% in the "worst" categories). We propose to address this issue by a simple, iterative self-verification process, which gradually shifts unverified labels into the verified, trusted training set. The decision criterion for self-verifying a training example is the prediction consensus of a previous snapshot of the network on multiple short sliding window excerpts of the training example at hand. On the unseen test data, an ensemble of three networks trained with this self-verification approach achieves a mean average precision (MAP@3) of 0.951. This is the second best out of 558 submissions to the corresponding Kaggle challenge.

Index Terms— Audio-tagging, Fully Convolutional Neural Networks, Noisy Labels, Label Self-Verification

1. INTRODUCTION

This short paper describes our approach[1] to the first "Freesound General-purpose Audio Tagging Challenge" which is carried out as Task 2 of the DCASE 2018 Challenge [1]. The central motivation for this challenge is to foster research towards more general machine listening systems capable of recognizing and discerning a wide range of acoustic events and audio scenes.

In particular, we aim at building an audio tagging system which assigns one out of 41 potential candidate labels to an unknown audio recording of arbitrary length. The labels comprise sound events such as music instruments, human sounds, animals, or domestic sounds [1]. What makes working with this data challenging is twofold: Firstly, the data set is collected from Freesound[2], which is a repository for user-generated audio recordings capturing diverse content with highly varying signal lengths recorded under diverse conditions [2, 3]. Secondly, the development (or training) dataset is delivered only partly with manually annotated ground truth labels. For the remaining recordings the labels are automatically generated and comprise up to 30-35% label noise in the "worst" categories. In the remainder of this paper, we refer to the manually annotated training observations as *verified* and to the additionally automatically annotated observations as *unverified*.

The central idea of our approach is to address the problem of noisy labels by training a fully convolutional audio classification network, which is iteratively fine-tuned by self-verifying the parts of the training observations provided without manually verified ground truth.

Technically, the task at hand is similar to other tasks (of earlier versions) of the DCASE challenge, which focus on detection or classification of audio scenes [4]. Therefore also our network architectures are inspired by earlier works addressing these problems [5]. Naturally, a large number of similar architectures and neural networks have proven to be powerful tools in this setting [6, 7, 8, 9, 10, 11]. There is also a large amount of prior work on training neural networks in the presence of label noise (see e.g. [12, 13, 14]). Due to the specifics of the current task (the labels of a large fraction of the data have verified manually and thus can be trusted), we opt for a straight forward iterative self-verification strategy. More concretely, to verify possibly noisy labels, our approach compares the labels of unverified examples with the predictions of a neural network, i.e. this can be interpreted as a version of "supervised" pseudo labeling [15]. A related approach addressing noisy labels by active label correction has been recently proposed in [16].

2. AUDIO DATA PREPROCESSING

Before we present the data to our networks, we apply several audio preprocessing steps including silence clipping and spectrogram computation.

The first step in our pipeline is normalizing the audio signal to a dB-level of -0.1. Next, we clip potential silence in the beginning and the end of the normalized audio using *SoX*[3]. This step is important as we later on optimize our networks on short sliding windows of the original files and want to avoid presenting training observations containing only silence along with the original label of the file[4]. Figure 1 shows an audio example of class *Knock* where this preprocessing step has a severe impact, reducing the effective length of the spectrogram from 435 to only 186 frames. Note that the part of the spectrogram covering actual audio content is preserved.

[1] Code: `https://github.com/CPJKU/dcase_task2`
[2] `https://freesound.org/`
[3] `http://sox.sourceforge.net/`
[4] For further details on how the audio signals are preprocessed we refer to our write-up: `https://cpjku.github.io/dcase_task2/`

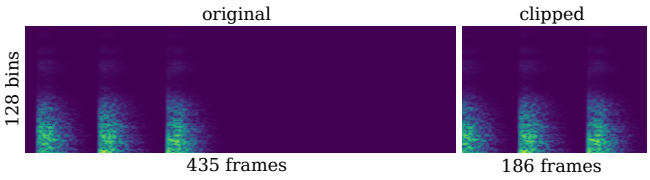


Figure 1: Log-Spectrogram (Version-2) of audio signal of class *Knock* before and after silence clipping.

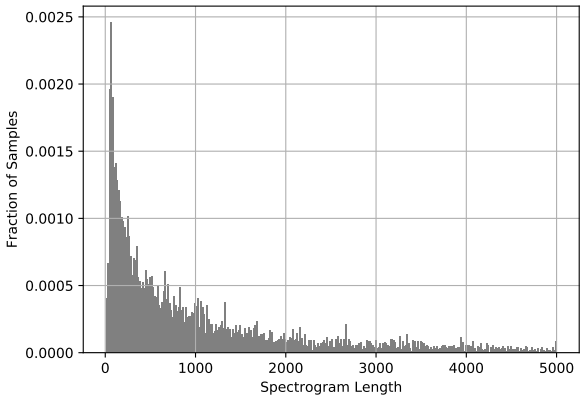


Figure 2: Distribution of spectrogram lengths computed with spectrogram *version-1*.

Before computing the spectrograms we resample the audio signals to 32,000 Hz, and compute a Short Time Fourier Transform (STFT) using 1024-sample hann windows. To try to capture different aspects of the audio, we extract two different spectrogram versions for our final submission:

Version-1 uses an STFT hop-size of 192 samples. Given this spectrogram, we apply a perceptual weighting to the individual frequency bands of the power spectrogram [17][5]. Finally, we apply a mel-scaled filterbank yielding 128 frequency bins per data point.

Version-2 uses an STFT hop-size of 128 samples and does not apply perceptual weighting but takes the logarithm of the power spectrogram instead. Finally it is post-processed with a logarithmic filter-bank again resulting in 128 frequency bins [18].

An additional characteristic of the data at hand is the varying length of the recordings. Figure 2 shows the distribution of spectrogram lengths for spectrogram *version-1*. As convolutional neural networks – which are the central component of our method – in general have a fixed field of view (input dimensions), it is desirable to work with audio excerpts consisting of the same number of frames. Additionally, to design convolution networks including max-pooling layers with a certain depth, we need to exceed a minimum input dimensionality[6]. To that end, we fix a target length of 3000 frames and simply repeat a given excerpt in case it is too short and clip it at 3000 frames in case it is too long. The 3000 frame threshold is chosen intuitively as the spectrogram length distribution in Figure 2 has a long tail with few observations.

[5] `librosa.core.perceptual_weighting`

[6] After each max-pooling layer the dimensionality of the input / feature maps gets halved. This of course restricts the maximum depth of a network.

Input $1 \times 384 \times 128$
5×5 Conv(pad-2, stride-2)-64-BN-ReLU
3×3 Conv(pad-1, stride-1)-64-BN-ReLU
2×2 Max-Pooling + Drop-Out(0.3)
3×3 Conv(pad-1, stride-1)-128-BN-ReLU
3×3 Conv(pad-1, stride-1)-128-BN-ReLU
2×2 Max-Pooling + Drop-Out(0.3)
3×3 Conv(pad-1, stride-1)-256-BN-ReLU
Drop-Out(0.3)
3×3 Conv(pad-1, stride-1)-256-BN-ReLU
Drop-Out(0.3)
3×3 Conv(pad-1, stride-1)-384-BN-ReLU
Drop-Out(0.3)
3×3 Conv(pad-1, stride-1)-384-BN-ReLU
2×2 Max-Pooling + Drop-Out(0.3)
3×3 Conv(pad-1, stride-1)-512-BN-ReLU
3×3 Conv(pad-1, stride-1)-512-BN-ReLU
1×2 Max-Pooling + Drop-Out(0.3)
3×3 Conv(pad-1, stride-1)-512-BN-ReLU
3×3 Conv(pad-1, stride-1)-512-BN-ReLU
1×2 Max-Pooling + Drop-Out(0.3)
3×3 Conv(pad-0, stride-1)-512-BN-ReLU
Drop-Out(0.5)
1×1 Conv(pad-0, stride-1)-512-BN-ReLU
Drop-Out(0.5)
1×1 Conv(pad-0, stride-1)-41-BN
Global-Average-Pooling
41-way Soft-Max

Table 1: Network architecture. BN: Batch Normalization, ReLU: Rectified Linear Unit.

3. NETWORK AND TRAINING DETAILS

In this section, we describe the neural network architectures as well as the optimization strategies used for training our audio scene classification networks.

3.1. Network Architecture

Our basic network architecture is a fully convolutional neural network as depicted in Table 1. In total we use three slightly modified versions of this architecture for our submission, but the general design principles remain the same. The feature learning part of our model follows a *VGG style network* [19], and the classification part of the network is designed as a global average pooling layer [20] over 41 feature maps (one for each class) followed by a *softmax* activation. In our experiments, global average pooling over per-class feature maps consistently outperforms networks using fully connected layers as a classification output. As an activation function within the network we use Rectified Linear Units (ReLUs) in combination with batch normalization [21]. Overall, this model comprises 14,865,124 trainable parameters.

3.2. General Training Procedure

At training time, we show the network randomly selected 384 frame excerpts of the full spectrograms, while presenting the whole 3,000 frame spectrograms during testing. This is technically possible as

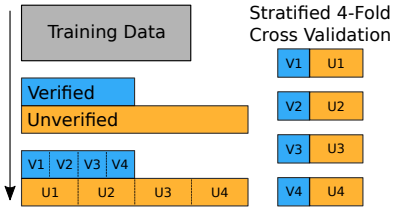


Figure 3: Stratified 4-fold cross-validation setup. After splitting into verified and unverified data we create eight stratified splits, each preserving the original label distribution. As a last step we assemble the eight sub-splits into four distinct folds.

our network is fully convolutional and can therefore process audio excerpts of varying length. Intuitively, presenting shorter excerpts for training should mitigate the effect of over-fitting to the individual training examples, as we end up with a much larger amount of shorter sub-excerpts. To further prevent over-fitting, we additionally apply mixup-data augmentation [22] with an $alpha$ of 0.3.

As optimizer we use the ADAM update rule [23] with an initial learning rate of 0.001 and a mini-batch size of 100 samples for training the initial version of our models. Each model is trained for 500 epochs, where we linearly decay the learning rate to zero starting from epoch 100. When fine tuning our models with iterative self-verification (see Section 4), we use a slightly modified training setup as described below.

4. 4-FOLD ITERATIVE SELF-VERIFICATION

This section describes our iterative self-verification loop to address the noisy labels in the development dataset. The central idea is to gradually shift unverified labels into the verified, trusted training set for fine-tuning the models.

4.1. 4-Fold Cross-Validation Setup

One crucial component of our self-verification approach is the way we prepare our training-validation-setup. To enable the proposed step-wise verification approach, we have to design our folds in a way that parts of the data (which we would like to verify) are never presented to the verification model for training. Otherwise, the neural network would learn the entire training set by heart, even in the presence of noisy labels [24], and its prediction become worthless for the verification strategy. Figure 3 provides an overview on how we prepare our split. First, we separate the development dataset into verified and unverified observations. Second, we split each of the two subsets into four stratified sub-folds, meaning that the label distribution in the sub-folds remains the same as in the original dataset. We consider this an important detail as the challenge organizers state on the official web page that the unseen test data exhibits a similar label distribution. In fact, for selecting our final submission we did not rely on the public Kaggle-Leaderboard but on the average performance on our local 4-fold cross-validation setup. Each of our local validation folds contains approximately 900 verified files which is substantially more than the 300 files (19% of official test data) considered for the public leaderboard. Although preparing this validation setup is straight forward, it is a crucial step as it is the basis for our self-verification pipeline described below.

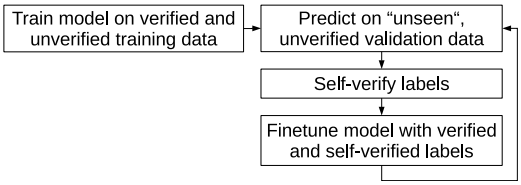


Figure 4: Iterative self-verification loop and model fine-tuning.

4.2. Iterative Self-Verification Loop

Figure 4 provides an overview of our iterative self-verification loop. Step one of this procedure is to train an initial model utilizing all the training data of a fold also including the unverified samples. We train one model for each fold as described in Section 3.2 and keep the best parametrization according to its validation loss on the verified validation data excluded from training. Note that the unverified data is not considered for model selection. This is also the main reason why we provide separate stratified splits for verified and unverified observations as it should provide us with an as reliable estimate of the real model performance as possible.

Once the initial model is trained we use it to predict the labels of its respective unseen validation examples. However, in contrast to the model selection we now only predict on the unverified observations. In particular, we draw K random 384 frame excerpts of the original 3000 frame spectrograms of the audio clip to verify. We then compute the average of the individual posterior class distributions $p_i(y|x)$ of these K excerpts:

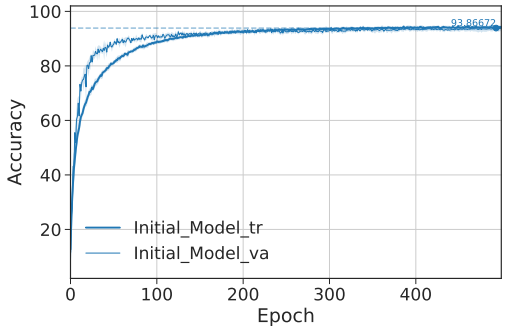
$$\bar{p}(y|x) = \frac{1}{K}\sum_{i=1}^{K} p_i(y|x) \quad (1)$$

with $K = 25$, $x \in \mathbb{R}^{384\times128}$ and $y \in \{0, ..., 40\}$. We then proceed by considering unverified examples as correctly annotated if:

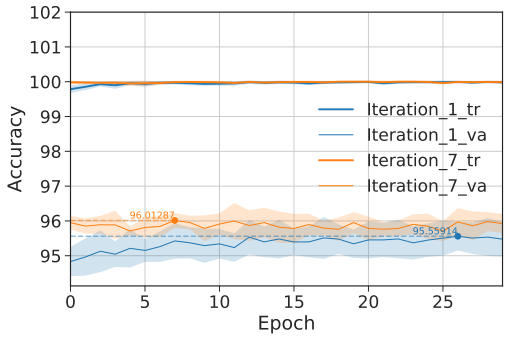
1. The provided unverified label y_u matches the label $\underset{ya}{\operatorname{argmax}}\ \bar{p}(y|x)$ predicted by the average of the individual posterior distributions.
2. The average of the target class posteriors exceeds 0.95.
3. A maximum count of 40 self-verified examples per class is not yet reached.

The intuition behind this approach is that especially for the unverified training examples multiple different classes might be present in a single audio recording. Still it is annotated with a single and hence unreliable label. When predicting on multiple random sub-excerpts of the recording this should be revealed by exhibiting a low average posterior probability $\bar{p}(y|x)$ for the provided target class label. The last condition for self-verification is introduced as some classes have very few examples and we want to avoid shifting the original label distribution of the dataset. As our procedure is based on four distinct cross-validation folds each unverified example is considered for verification once per iteration.

The final stage of this self-verification loop is to fine-tune the four initial fold models, this time using the officially provided verified observations and the ones passing the self-verification conditions in the previous stage. For the fine-tuning stage we train for 30 epochs with an initial learning rate of 0.0002, which is again decayed to zero starting after five epochs. We do not use mixup



(a) Initial model trained on entire dataset.



(b) Models after self-verification fine-tuning.

Figure 5: Comparison of model performance of network trained on (a) all provided data including noisy labels and (b) all data which passed the proposed self-verification at the respective iteration. We report mean and standard deviation (shaded area) of the classification accuracy on training (tr) and validation (va) set averaged across our four development folds. For the present case self-verification iteration 7 yields the best model (fine-tuned for 7 $\times$ 30 epochs).

data augmentation in this stage anymore. After fine-tuning we go back to step two and repeat the whole procedure in a loop for ten times. Once the ten iterations are completed, we select the model parameterization of the iteration showing the lowest average validation loss on the officially verified validation data. Note that we do not reset the network to the initial parameterization after each fine-tuning iteration but continue training the same model.

5. EXPERIMENTAL RESULTS

In this section we report our empirical results on both, our local validation setup as well as the public and private Kaggle-Leaderboard. As evaluation measures we consider the Mean Average Precision @ 3 (MAP@3), which is the score officially used for the challenge, as well as classification accuracy and F-Score when presenting the performance on individual classes.

Figure 5a shows training and validation accuracy on our local 4-fold setup of the model described in Table 1 when training on the entire dataset including unverified labels. This corresponds to the first stage of the self-verification procedure in Figure 4. We report the mean and standard deviation of the network across the four distinct folds where the best average accuracy on the validation set after stage one is 93.87%. In Figure 5b we show the performance of the same model after the first and seventh fine-tuning iteration.

	Public	Private	Public & Private
MAP@3	0.9563	0.9518	0.9526
Accuracy	93.688	92.610	92.812

Table 2: Audio tagging performance on the *public*, *private*, and *public & private* test set (Kaggle-Leaderboard).

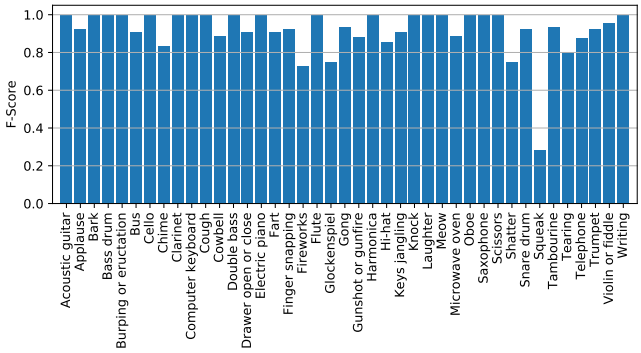


Figure 6: F-score of final submission on individual classes.

Already after the first self-verification iteration, we observe a performance improvement to 95.56%. Finally, the model reaches its best performance in iteration seven with an average verified validation set performance of 96.01%.

Before preparing our submission for the challenge, we trained three similar networks to the one in Table 1 (two on spectrogram *version-1* and one on spectrogram *version-2*) and averaged their predictions. When evaluating this submission on the official test data set we achieve the second best scoring submission in the final private Kaggle-Leaderboard with a MAP@3 of 0.9518. For easier comparability with future research on this dataset, we also report detailed results on the different subsets of the test data in Table 2.

To provide an intuition on how well the approach performs on individual classes, we report the F-score for all 41 classes in Figure 6. We observe that the model achieves an F-score above 0.8 for all classes except for *Squeak*, *Fireworks*, and *Glockenspiel*. Many of the classes are even recognized with a perfect score of 1.0. Considering the noisy labels and that there are 41 different classes to distinguish we take this as a remarkable result.

6. CONCLUSION

In this workshop paper, we described our submission to the first Freesound general-purpose audio tagging challenge carried out with DCASE 2018. Our proposed approach is an iterative self-verification loop built on top of a fully convolutional neural network. After an initial training stage using all the data, we iteratively fine-tune our networks using label self-verification. The central idea of our proposal is to add unverified examples step-by-step to the training set based on the prediction consensus of our networks with the suggested, noisy labels. For a single model, this approach improves the classification accuracy from 93.87% to 96.01% on the local validation set. When training an ensemble of three similar networks in this fashion, we are able to achieve a MAP@3 of 0.9518 (92.610% accuracy) on the final private Kaggle-Leaderboard. Overall this yields the second best scoring out of 558 submissions.

7. REFERENCES

[1] E. Fonseca, M. Plakal, F. Font, D. P. W. Ellis, X. Favory, J. Pons, and X. Serra, "General-purpose tagging of freesound audio with audioset labels: Task description, dataset, and baseline," 2018, submitted to DCASE2018 Workshop. [Online]. Available: https://arxiv.org/abs/1807.09902

[2] E. Fonseca, J. Pons, X. Favory, F. Font, D. Bogdanov, A. Ferraro, S. Oramas, A. Porter, and X. Serra, "Freesound datasets: A platform for the creation of open audio datasets," in *Proceedings of the 18th International Society for Music Information (ISMIR)*, Suzhou, China, 2017, pp. 486–493.

[3] J. F. Gemmeke, D. P. W. Ellis, D. Freedman, A. Jansen, W. Lawrence, R. C. Moore, M. Plakal, and M. Ritter, "Audio set: An ontology and human-labeled dataset for audio events," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, New Orleans, USA, 2017, pp. 776–780.

[4] A. Mesaros, T. Heittola, and T. Virtanen, "A multi-device dataset for urban acoustic scene classification," 2018, submitted to DCASE2018 Workshop. [Online]. Available: https://arxiv.org/abs/1807.09840

[5] H. Eghbal-Zadeh, B. Lehner, M. Dorfer, and G. Widmer, "Cp-jku submissions for dcase-2016: A hybrid approach using binaural i-vectors and deep convolutional neural networks," *IEEE AASP Challenge on Detection and Classification of Acoustic Scenes and Events (DCASE)*, 2016.

[6] S. Hershey, S. Chaudhuri, D. P. W. Ellis, J. F. Gemmeke, A. Jansen, R. C. Moore, M. Plakal, D. Platt, R. A. Saurous, B. Seybold, M. Slaney, R. J. Weiss, and K. W. Wilson, "CNN architectures for large-scale audio classification," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, New Orleans, USA, 2017, pp. 131–135.

[7] K. J. Piczak, "Environmental sound classification with convolutional neural networks," in *International Workshop on Machine Learning for Signal Processing (MLSP)*, Boston, USA, 2015, pp. 1–6.

[8] J. Salamon and J. P. Bello, "Deep convolutional neural networks and data augmentation for environmental sound classification," *IEEE Signal Processing Letters*, vol. 24, no. 3, pp. 279–283, 2017.

[9] T. H. Vu and J.-C. Wang, "Acoustic scene and event recognition using recurrent neural networks," *Detection and Classification of Acoustic Scenes and Events*, vol. 2016, 2016.

[10] K. Choi, G. Fazekas, and M. B. Sandler, "Automatic tagging using deep convolutional neural networks," in *Proceedings of the 17th International Society for Music Information Retrieval Conference (ISMIR)*, New York City, United States, 2016, pp. 805–811.

[11] Y. Xu, Q. Kong, Q. Huang, W. Wang, and M. D. Plumbley, "Convolutional gated recurrent neural network incorporating spatial features for audio tagging," in *2017 International Joint Conference on Neural Networks (IJCNN)*, Anchorage, USA, 2017, pp. 3461–3466.

[12] S. Sukhbaatar and R. Fergus, "Learning from noisy labels with deep neural networks," *CoRR*, vol. abs/1406.2080, 2014.

[13] S. E. Reed, H. Lee, D. Anguelov, C. Szegedy, D. Erhan, and A. Rabinovich, "Training deep neural networks on noisy labels with bootstrapping," *CoRR*, vol. abs/1412.6596, 2014.

[14] L. Jiang, Z. Zhou, T. Leung, L. Li, and L. Fei-Fei, "Mentornet: Learning data-driven curriculum for very deep neural networks on corrupted labels," in *Proceedings of the 35th International Conference on Machine Learning (ICML)*, Stockholm, Sweden, 2018, pp. 2309–2318.

[15] D.-H. Lee, "Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks," in *Workshop on Challenges in Representation Learning, ICML*, vol. 3, 2013, p. 2.

[16] J. Kremer, F. Sha, and C. Igel, "Robust active label correction," in *International Conference on Artificial Intelligence and Statistics (AISTATS)*, Playa Blanca, Spain, 2018, pp. 308–316.

[17] B. McFee, C. Raffel, D. Liang, D. P. Ellis, M. McVicar, E. Battenberg, and O. Nieto, "librosa: Audio and music signal analysis in python," in *Proceedings of the 14th python in science conference*, 2015, pp. 18–25.

[18] S. Böck, F. Korzeniowski, J. Schlüter, F. Krebs, and G. Widmer, "Madmom: A new python audio and music signal processing library," in *Proceedings of the 2016 ACM on Multimedia Conference*, Amsterdam, The Netherlands, 2016, pp. 1174–1178.

[19] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proceedings of the International Conference on Learning Representations (ICLR)*, San Diego, USA, 2015.

[20] M. Lin, Q. Chen, and S. Yan, "Network in network," in *Proceedings of the International Conference on Learning Representations (ICLR)*, Banff, Canada, 2014.

[21] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, Lille, France, 2015, pp. 448–456.

[22] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," *arXiv preprint arXiv:1710.09412*, 2017.

[23] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proceedings of the International Conference on Learning Representations (ICLR)*, San Diego, USA, 2015.

[24] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, "Understanding deep learning requires rethinking generalization," in *In Proceedings of the 5th International Conference on Learning Representations (ICLR)*, Toulon, France, 2016.

AN EXTENSIBLE CLUSTER-GRAPH TAXONOMY FOR OPEN SET SOUND SCENE ANALYSIS

Helen L. Bear and Emmanouil Benetos

School of Electronic Engineering and Computer Science, Queen Mary University of London, UK
{h.bear, emmanouil.benetos}@qmul.ac.uk

ABSTRACT

We present a new extensible and divisible taxonomy for open set sound scene analysis. This new model allows complex scene analysis with tangible descriptors and perception labels. Its novel structure is a cluster graph such that each cluster (or subset) can stand alone for targeted analyses such as office sound event detection, whilst maintaining integrity over the whole graph (superset) of labels. The key design benefit is its extensibility as new labels are needed during new data capture. Furthermore, datasets which use the same taxonomy are easily augmented, saving future data collection effort. We balance the details needed for complex scene analysis with avoiding 'the taxonomy of everything' with our framework to ensure no duplicity in the superset of labels and demonstrate this with DCASE challenge classifications.

***Index Terms*—** Taxonomy, ontology, sound scenes, sound events, sound scene analysis, open set

1. INTRODUCTION

In sound scene analysis, that is describing a scene and its constituent events from an audio input, most work poses the problem as a closed set problem [1]. This means researchers use a defined set of class labels with various levels of confidence. In doing so, many datasets and associated taxonomies/ontologies have been created [2, 3, 4]. These approaches are based on the assumption that sound scenes can be described from a finite collection of labels. However, particularly for complex real-world scenes, the problem is more akin to an open set classification task [5, 6]. That is, there are infinite possible descriptor labels, and furthermore, many combinations of labels possible. The range of tasks in computational sound scene analysis is varied (e.g. scene classification, event recognition) and each time a new dataset is created, we also create new sets of labels. This reduces the reusability and value of data which is expensive and time-consuming to collect and annotate. Therefore, we seek a new structure taxonomy to support the research community, and thus we need a class labeling mechanism which can:

- extend as the complexity of scenes develops and research spreads into new scenes of interest,
- not duplicate descriptors across sub-areas of a taxonomy,
- be divided up for tackling nuanced sub-problems in sound scene analysis, and
- enable multi-perspective descriptors; that is, how a human perceives a scene rather than physical logical descriptors.

This work was funded under EPSRC grant EP/R01891X/1. EB is supported by a RAEng Research Fellowship (RF/128).

This paper provides a framework for a modifiable and extensible sound scene taxonomy for *all* scenes, where one can analyse a set of any events in any environment. The result is scenes that are describable by a set of descriptors consistent across datasets. Descriptors are singular scene labels which can describe: the environment, events, or the context (how a human could perceive the scene).

The rest of this paper is structured as follows: we discuss the background of taxonomy development, before presenting our extensible taxonomy architecture and a framework for populating it with labels and maintaining it. We then demonstrate it using the label sets provided with the DCASE challenge label sets from 2013–2018 before summarising the benefits of this new approach.

2. BACKGROUND

We begin with the following definitions:

a **taxonomy** is *a scheme of classification*,

a **label set** is *a collection of class names*,

an **ontology** is *a set of concepts and categories in a domain that shows their properties and the relations between them* and,

a **thesaurus** is *a book that lists words in groups of synonyms and related concepts*.

Our proposal is an extensible taxonomy which combines an ontology to organise label sets supported by a thesaurus to avoid duplication or misnomers between related subsets of labels. In doing so, we organise a collection of label sets into a graph structure to enable relations between class labels and subsets of labels. This moves us away from using hierarchies to organise classification labels. Benefits of doing so allows a user to have specificity and precision at multiple levels for both scenes and events, and researchers can simply share lists of edges formed by label pairs to share the whole taxonomy or part required thereof.

To do this, we build upon prior taxonomy work in sound scene analysis. We first discuss soundscapes, then events as these are typically (not always) addressed independently. Our third section reviews joint attempts. As we discuss this prior work, we recall that the purpose of sound scene analysis can vary. The requirements of a taxonomy which is fit for urban events, are unlikely to be suitable for urban scenes without some processing or modification. Therefore, as we discuss previous works we focus on their primary goal, before addressing modification requirements to aid an extensible taxonomy for complex scenes.

Soundnet [7] was developed using transfer learning from computer vision research. Scene understanding is a major computer vision research topic including for example object recognition and semantic understanding. Using this prior knowledge, transfer learning enabled deduction of acoustic labels. *AudioSet* [3] is a two tier hierarchical ontology of 632 audio classes based on prior literature and using youtube video clips. *Urban sounds* [2] groups with four top

level groups: human, nature, mechanical and music. Here the authors target urban soundscapes and determine that leaf labels must be specific, e.g. car "brakes", "engine" or "horn", instead of simply "car". However, this means that the leaf labels are specific for the clusters and cannot be shared or compounded with others for complex scenes outside the remit of urban environments.

Gaver's taxonomy [8] was evaluated by Houix [9] and found that sounds can be cross-classified by alternative principles depending on sound presentation and the background of a listener. This is supported by Guyot [10] who observed a distinction in the classification labels used between acoustician/non-acoustician listeners. In further human classification experiments, Van der Veer [11] and Shubert [12] demonstrated that when classifying sounds, humans tend to use compounds of objects and actions. This is evidence that a wholly encompassing open set taxonomy which permits compound descriptors from fundamental labels would be beneficial for fair audio assessment. One approach in [13] uses a hierarchy between the environment and the scene. The classification strategy uses context (environment by our definition) to aid event detection similar to human comprehension methods. Within their own dataset, environments are distinct, e.g. beach, park, on a bus. In addition, Pijanowski *et al.* [14] used three abstract groupings; geophony, biophony, and anthophony for environmental/outside spaces. Similarly, Delage in [15] used three abstract groups for urban sounds: sounds of nature, direct human activity, and indirect human activity. These are examples of clusters/groups of labels within a whole set and we use this model as inspiration for our cluster-graph ontology (as we present in Sec. 3).

Of these taxonomies, many use abstract labels for grouping types of labels. Yet, the labels in these groups can duplicate across groupings which creates confusion across datasets and research problems. Homographs *(def: each of two or more words having the same spelling or pronunciation but different meanings and origins)* and synonyms *(def: a word or phrase that means exactly or nearly the same as another word or phrase in the same language)* [16] are troublemakers in developing taxonomies for complex sound scenes. Compounded with the 'extra' abstract labels, we witness inaccuracies and ambiguous classifications. Therefore, we need a set of rules for extending the taxonomy for each cluster in the graph. Some prior work does not restrict itself to only Urban/Rural sounds or Scene/Event sound schemes. A number of schemes have wide reaching coverage, namely [10, 13, 8, 9]. Importantly, not all taxonomies fit into this simple cross-referencing structure, thus any future solution must enable cross problem label sharing for complex scenes. Also, any new taxonomy should address the human ability to 'cross-classify', i.e. enabling a machine to classify scenes and their components in multiple variants. That is, one class might exist in multiple different scenes, and the best approach for recognising that class can vary by the scene or context.

The first thesaurus: Wordnet [17] is an open-source online thesaurus containing groups of synonyms named *synsets*. Synsets are indexed by the preferred term, that is the word most commonly used. Using this we can ensure that our cluster-graph has one single label per leaf and that highest usage synonyms are used to reduce duplicity and to manage homographs. All punctuation should be removed, and labels saved in lower case.

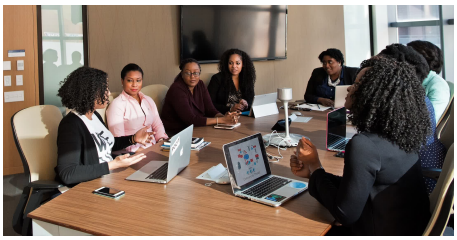
3. A GRAPH TAXONOMY

In Sec. 2 we stated our proposal is an extensible taxonomy which combines a graph-based ontology to organise label sets supported by a thesaurus to avoid duplication or misnomers between related subsets of labels. This means that the structure and relations between class labels are modelled as a graph, where class labels are grouped in clusters of leaf nodes, label sets are subgraphs, and the relations between these are edges. As we build upon label sets already provided, this will be a cluster graph [18]. This structure is new for sound scene analysis taxonomy, although graphs have been used in vision tasks [19].

The graph taxonomy enables one to make cuts through the total set of all labels/leaf nodes such that it can be sliced into relevant subsets of labels for different challenges in sound scene analysis. Each cluster is related via the graph schema so enables use of the same categories presented in prior datasets (reorganised) into an cluster graph. This also encourages data augmentation (e.g. [20]) across clusters so that researchers can quickly amalgamate bigger datasets without the collection and annotation costs. Depending on raw data used, intra-cluster distance measures can be calculated for multi-label classification methods.

A significant benefit of this architecture is that it can be represented in both graph and set form, more commonly known as a Venn diagram [21]. With both we can show both the detail of the graph and the paths between labels, but also the most common ways of cutting the graph (based on prior works) into subsets, or sub-graphs.

In this taxonomy we use the following terms as defined:
Environment: tangible description of a scene, e.g. indoors, light, building, people.
Context: a human perceptual description of a scene, e.g. meeting, party, sea side.
Event: sound emitting actions or objects in the scene, e.g. speech, writing, keyboard presses, walking.



Example Scene

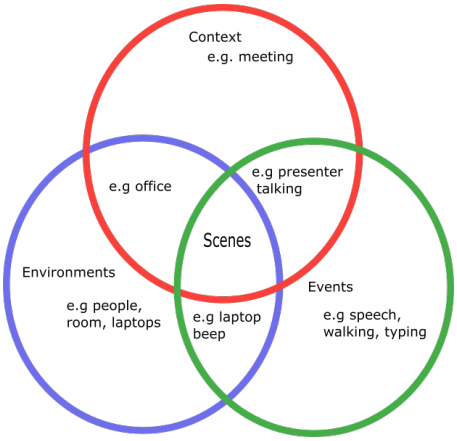


Figure 1: Example scene (top) [22] and first graph cuts for subsets of label sets (bottom).

These terms form the fundamental ontology of label sets as shown in Fig. 1. This Venn diagram shows boundaries between the label subsets (labels and the subgroupings are graph nodes), and how the subsets relate to each other to form alternative descriptors. The example labels are examples for a scene similar to Fig. 1(top).

Thus we begin with a superset T, which is a set of all labels, whether part of the environment en, event ev, or context c subsets. At this top level, all labels in T can be related to all others which forms the full graph $G = (T, E_t)$ where E_t is the set of edges between all pairs of T. With T, many subsets t of T can be created for any sound scene problem (urban, office events etc), with the reassurance that a consistent T enables extension of current taxonomies.

3.1. The ontology O

The remaining element of this taxonomy is forming an ontology of the relations between subsets. We observed in prior work both the use of compound labels and also the human tendency to describe sound scenes perceptually. The compound labels form what is seen in Figure 1 as overlap subsets between the event and environment subsets, and the human perception labels are the context subset.

Collectively, these subsets are the abstract classes used in prior taxonomies but now we can also use them as subset labels to support multi-label analysis without needing data annotated at the label level in the leaf nodes, reducing the annotation overhead and maximising usability of the data already available.

Subset names are **not** included as nodes (labels) in the graph. These are maintained as a separate list. It is essential that there is no duplication over all subsets, that is the same label can appear in more than one subset, but only as itself, not a synonym. It is also essential that T contains all unique elements of all subsets. This constrains T to prevent duplicity and unnecessary complexity.

3.2. Sub graph architecture

Because our ontology architecture is a graph, we have the significant benefit that we can make many graph cuts through the total set of labels (T) to create new sub-graphs, or new label sets (as we will see in Sec. 5). The obvious subgroups are the pre-populated clusters building on the label sets already available in literature. Therefore, we avoid using abstract names for labels. Rather, these should be a suitable name for an aggregate set, incorporating the super/sub class relation from a hierarchy of previous taxonomies without creating a node identifier that is not suitable as a label, e.g. 'subway train' is a composite of 'subway' and 'train' which stand alone as labels.

Different complex, polyphonic sound scenes require variable numbers of categories to describe the scene. Using our Fig 1 example, some would relate to the environment e.g. 'office', some events (or actions) like 'talking', and some objects, 'laptops'. Given the volume of possible combinations, we suggest using label vectors to represent a set of categories per scene or compound labels for measuring accuracy, rather than creating new ones.

The use of context to improve event classification has been seen as beneficial in sound scene analysis [13]; for example, if we are confident the scene is a beach, we raise the probability of events being waves or walking on sand. An alternative example is, the lack of an adverse class infers a class to be true, meaning if it is not raining nor windy then the probability of a sunny scene is more likely. This is where the clusters of label sets need weighted relations between labels within a subset or, relations (again weighted) between sets of labels. Any weights would be subject to data selection.

Each cluster can be treated as both its own ontology, that is each label in a cluster set will relate to the other labels in the cluster, as such, distances between the labels might be calculated. Furthermore, as the taxonomy extends, distances between clusters, or pairs of leaves in different clusters, can be used to improve the confidence of multi-label classification predictions.

4. NEW LABEL FRAMEWORK

In order to add to our graph, we present a framework (Fig. 2) to maintain graph integrity. We use this with the DCASE challenge labels to initialise T.

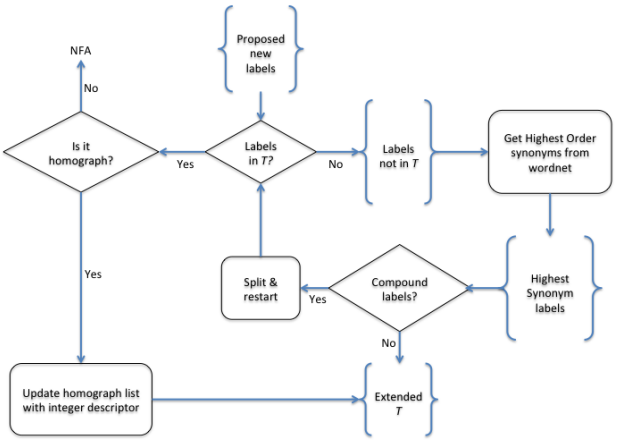


Figure 2: Framework for extending the taxonomy.

5. INITIALISATION WITH DCASE EXAMPLES

We use all the label sets from DCASE event challenges 2013 to 2018 [4, 23, 24, 25] as listed chronologically from left to right in Table 1, and our framework to demonstrate how these produced T [1]. We denote each set in the format 'DxxTy' meaning D=DCASE, xx=the year, T=Task, and y=the task number. Working from left to right in Table 1, the labels for D16T2 are all either identical ('clearing throat', 'door knock', 'door slam', 'drawer', 'phone ringing','speech') or homographs and synonyms ('cough'='coughing', 'laughter'='human laughter', 'keyboard clicks'='keyboard', 'keys clinging'='keys', 'turning page'='page turning') of Events in D13T2 and T3 so we do not reuse them. This process gives us $Ev_0 = \{doorknock, doorslam, speech, laughter, keyboard, impact, keys, phone, ringing, turning, page, cough, printer, alert, beep, short, throat, clear, mouse, click, drawer, switch\}$.

Using the same process on D16T3 we produce {Rustling, snapping, cupboard, cutlery, dishes, impact} and some compound labels, {glass jingling, people walking, washing dishes, and water tap running}. The framework dismantles the compound labels into {glass (obj), jingling (act), people (obj), walking (act), washing (act), we omit 'dishes' as a duplicate, tap (obj), and running act}. $Ev_1 = Ev_0 + \{Rustling, snapping, cupboard, cutlery, dishes, impact, glass, jingling, people, walking, washing, tap, running\}$.

[1] T and the first subsets: en, ev, c are available from `soundscape.eecs.qmul.ac.uk/the-extensible-taxonomy/`

Table 1: Event label set clusters from various DCASE challenges (2013,2016-2018)

DCASE 13' (T2&3)	DCASE 16'(T2)	DCASE 16'(T3)	DCASE 17'(T3)	DCASE 17'(T4)	DCASE 18' (T4)
door knock,	clearing throat	(object) rustling	brakes squeaking	train horn	speech
door slam,	coughing	(object) snapping	car	air horn, truck horn	dog
speech,	door knock	cupboard	children	car alarm	cat
laughter,	door slam	cutlery	large vehicle	reversing beeps	alarm
keyboard clicks,	drawer (close)	dishes	people speaking	ambulance (siren)	dishes
objects hitting table,	human laughter	drawer	people walking	police car (siren)	frying
keys clinging,	Keyboard	glass jingling		fire engine	blender
phone ringing,	keys (put on table)	object impact		civil defense siren	running water
turning page,	page turning	people walking		screaming	vacuum cleaner
cough,	phone ringing	washing dishes		bicycle	electric shaver/toothbrush
printer,	speech	water tap running		skateboard	
short alert-beeping,		(object) Banging		car	
clearing throat,		bird singing		car passing by	
mouse click,		car passing by		bus	
drawer		children shouting		truck	
switch		people speaking		motorcycle	
		people walking		train	
		wind blowing			

We continue in this fashion for all challenge label sets until we are left with: $T_{events} = \{door, knock, slam, speech, laughter, keyboard, impact, keys, phone, ringing, turning, page, cough, printer, alert, beep, short, throat, clear, mouse, click, drawer, switch, rustling, snapping, cupboard, cutlery, dishes, impact, glass, jingling, people, walking, washing, tap, running, brakes, car, squeaking, children, large, vehicle, bird, singing, passing, shouting, wind, blowing, train, horn, air, truck, reversing, siren, fire, police, ambulance, engine, screaming, bike, skateboard, motorbike, dog, cat, frying, blender, vacuum, cleaner, shaver, toothbrush\}$.

Table 2: Scene label set clusters from DCASE challenges (2013,2016-2018)

DCASE 13'(T1)	DCASE 16'&17' (T1)	DCASE 18' (T1)
busy street	bus	airport
quiet street	cafe / Restaurant	shopping mall
park	car	metro station
open-air market	city center	pedestrian street
bus	forest path	public square
subway-train	grocery store	street medium traffic
restaurant	home	tram (riding)
shop/supermarket	lakeside beach	bus (riding)
office	library	metro (riding)
subway station	metro station	urban Park
	office	
	residential area	
	train	
	tram	
	urban park	

In Table 2 we have listed label sets for each scene classification challenges from all DCASE workshops since 2013. Task 1 for DCASE 16 and DCASE 17 are identical therefore only listed once. The final set of labels produced with our framework which encompasses all previous labels is:
$T_{scenes} = \{bus, restaurant, shop, metro, airport, street, supermarket, quiet, busy, park, office, station, car, city, center, forest, pavement, library, train, tram, mall, public space, riding\}$.

Finalising T: we join T_{events} and T_{scenes} and remove duplicates between the two to form T. We further subselect the labels into the event, ev, environment, en, and context, c subsets. c is the smallest based on DCASE: $c = \{office, meeting, shopping\}$.

6. SUMMARY

In summary, with this extensible framework we hope to start an evolving taxonomy of classification labels for open set sound scene analysis. With this design, we enable machines to cross-classify as humans do; with consistent multiple taxonomies and using the cluster graph structure, this product enables correlation between dependent sound attributes in a scene, i.e. learning to discriminate the same event in different contexts. If we ensure that future dataset labeling strategies build upon those which already exist such as expanding this taxonomy, then we can amalgamate datasets for future research. We aim to use our framework to align other datasets such as AudioSet with the proposed taxonomy. Although our approach is a small overhead when annotating new datasets, the long term benefits of data augmentation outweigh the cost, and our framework is much simpler than using other data migration methods (e.g. [26]) used in reusing datasets.

Alongside sharing the initialised graph taxonomy, we have provided a central online point for links to future datasets which conform to the extensible approach so all researchers can link to the relevant parts/collections of datasets they wish to use for their own analyses[2]. A further benefit of this work is that it enables both bottom-up and top-down strategies for forming new label sets and sharing/combining them with other researchers. We hope that future DCASE challenge organisers will adopt this approach for managing labels in new datasets as this unification of sound labels will ease the annotation task.

[2]Email `casa.opentaxonomy@qmul.ac.uk` to add your dataset link

7. REFERENCES

[1] M. Aly, "Survey on multiclass classification methods," *Neural Networks*, vol. 19, pp. 1–9, 2005.

[2] J. Salamon, C. Jacoby, and J. P. Bello, "A dataset and taxonomy for urban sound research," in *22nd ACM international Conference on Multimedia*, 2014, pp. 1041–1044.

[3] J. F. Gemmeke, D. P. Ellis, D. Freedman, A. Jansen, W. Lawrence, R. C. Moore, M. Plakal, and M. Ritter, "Audio set: An ontology and human-labeled dataset for audio events," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017, pp. 776–780.

[4] D. Giannoulis, E. Benetos, D. Stowell, M. Rossignol, M. Lagrange, and M. D. Plumbley, "Detection and classification of acoustic scenes and events: An IEEE AASP challenge," in *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, 2013, pp. 1–4.

[5] D. Battaglino, L. Lepauloux, and N. Evans, "The open-set problem in acoustic scene classification," in *IEEE International Workshop on Acoustic Signal Enhancement (IWAENC)*, 2016, pp. 1–5.

[6] J. MacQueen *et al.*, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, vol. 1, no. 14, 1967, pp. 281–297.

[7] Y. Aytar, C. Vondrick, and A. Torralba, "Soundnet: Learning sound representations from unlabeled video," in *Advances in Neural Information Processing Systems*, 2016, pp. 892–900.

[8] W. W. Gaver, "What in the world do we hear?: An ecological approach to auditory event perception," *Ecological Psychology*, vol. 5, no. 1, pp. 1–29, 1993.

[9] O. Houix, G. Lemaitre, N. Misdariis, P. Susini, and I. Urdapilleta, "A lexical analysis of environmental sound categories." *Journal of Experimental Psychology: Applied*, vol. 18, no. 1, p. 52, 2012.

[10] F. Guyot, M. Castellengo, and B. Fabre, "A study of the categorization of a household noise corpus," in *Categorization and cognition: from perception to speech*, 1997, pp. 41–58.

[11] N. J. Vanderveer, *Ecological acoustics: Human perception of environmental sounds*. Unpublished dissertation, 1980.

[12] E. D. Schubert, *The role of auditory perception in language processing*. York, 1975.

[13] T. Heittola, A. Mesaros, A. Eronen, and T. Virtanen, "Context-dependent sound event detection," *EURASIP Journal on Audio, Speech, and Music Processing*, 2013.

[14] B. C. Pijanowski, A. Farina, S. H. Gage, S. L. Dumyahn, and B. L. Krause, "What is soundscape ecology? an introduction and overview of an emerging new science," *Landscape Ecology*, vol. 26, no. 9, pp. 1213–1232, 2011.

[15] P. Amphoux, "Paysage sonore urbain," *Le paysage et ses grilles, Franoise Chenet*, pp. 109–122, 1996.

[16] J. Simpson, E. S. Weiner, *et al.*, "Oxford English dictionary online," *Oxford: Clarendon Press. Retrieved March*, vol. 6, p. 2008, 1989.

[17] Princeton University, "About WordNet," 2010. [Online]. Available: http://wordnetweb.princeton.edu/perl/webwn

[18] D. B. West *et al.*, *Introduction to graph theory*. Prentice Hall Upper Saddle River, 2001, vol. 2.

[19] M. Sridhar, A. G. Cohn, and D. C. Hogg, "Discovering an event taxonomy from video using qualitative spatio-temporal graphs," in *19th European Conference on Artificial Intelligence (ECAI)*, vol. 215, 2010, pp. 1103–1104.

[20] Y. Bai, K. Yang, W.-Y. Ma, and T. Zhao, "Automatic dataset augmentation," *arXiv preprint arXiv:1708.08201*, 2017.

[21] H. B. Enderton, *Elements of set theory*. Academic Press, 1977.

[22] pexels.com, "Copyright free images," 2018. [Online]. Available: https://www.pexels.com/search/conference/

[23] A. Mesaros, T. Heittola, E. Benetos, P. Foster, M. Lagrange, T. Virtanen, and M. D. Plumbley, "Detection and classification of acoustic scenes and events: Outcome of the DCASE 2016 challenge," *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, vol. 26, no. 2, pp. 379–393, 2018.

[24] A. Mesaros, T. Heittola, A. Diment, B. Elizalde, A. Shah, E. Vincent, B. Raj, and T. Virtanen, "DCASE 2017 challenge setup: Tasks, datasets and baseline system," in *Workshop on Detection and Classification of Acoustic Scenes and Events (DCASE)*, 2017.

[25] "DCASE workshop and challenge," 2018. [Online]. Available: http://dcase.community/workshop2018/index

[26] S. H. Choe and Y. M. Ko, "Collective archiving of soundscapes in socio-cultural context," *iConference Proceedings*, 2015.

MULTI-LEVEL ATTENTION MODEL FOR WEAKLY SUPERVISED AUDIO CLASSIFICATION

Changsong Yu[1], *Karim Said Barsim*[1], *Qiuqiang Kong*[2], *Bin Yang*[1]

[1] Institute of Signal Processing and System Theory, University of Stuttgart, Germany
[2] Center for Vision, Speech and Signal Processing, University of Surrey, UK

ABSTRACT

In this paper, we propose a multi-level attention model for the weakly labelled audio classification problem. The objective of audio classification is to predict the presence or the absence of sound events in an audio clip. Recently, Google published a large scale weakly labelled AudioSet dataset containing 2 million audio clips with only the presence or the absence labels of the sound events, without the onset and offset time of the sound events. Previously proposed attention models only applied a single attention module on the last layer of a neural network which limited the capacity of the attention model. In this paper, we propose a multi-level attention model which consists of multiple attention modules applied on the intermediate neural network layers. The outputs of these attention modules are concatenated to a vector followed by a fully connected layer to obtain the final prediction of each class. Experiments show that the proposed multi-attention attention model achieves a state-of-the-art mean average precision (mAP) of 0.360, outperforming the single attention model and the Google baseline system of 0.327 and 0.314, respectively.

Index Terms— AudioSet, audio classification, attention model

1. INTRODUCTION

Audio classification aims to predict the presence or the absence of audio events in an audio clip. Audio classification has many applications such as multimedia information retrieval and public surveillance [1, 2]. Before 2017, datasets in audio processing are relatively smaller than datasets in computer vision such as ImageNet [3]. For example, UrbanSound dataset [4] consists of 27 hours of urban sound records with 3075 samples. ESC-50 dataset [5] consists of 2000 environmental recordings across 50 classes. The detection and classification of acoustic scenes and events (DCASE) challenge 2013, 2016, 2017 [1, 2, 6] datasets consists of several hours of data. Recently, Google published a large scale audio classification dataset called AudioSet [7] consists of 5,800 hours two million human-labeled 10-second audio clips covering 527 audio categories.

In AudioSet, each audio clip contains one or several labels, such as "cat", "speech" and "park" [7]. AudioSet is a *weakly labelled dataset* (WLD), that is, only the presence or the absence of sound events are known in an audio clip, without knowing the onset and offset time of the sound events. The duration of sound events in the WLD vary from milliseconds to seconds depending on the categories. For example, sound class such as "speech" can last a few seconds, while sound class such as "gunshot" only last for hundreds of milliseconds.

The audio classification problem with WLD is to design a system trained only on WLD. Many methods such as multiple instance learning (MIL) [8] has been used to solve the WLD audio classification [9] problem. In [10] a single-level attention model was proposed and outperformed both the MIL method [9] and the Google baseline deep neural network system [7] on AudioSet classification. This single-level attention model consists of three fully connected layers followed by an attention module. The motivation of the attention module is based on the observation that different segments in an audio clip contribute differently to the label of an audio clip. For example, the segments containing a sound event should be attended and the segments containing irrelevant noise should be ignored.

However, when using the single-level attention model, substantial information from the intermediate neural network layers is disregarded. Previous work [11, 12, 13] explored the features from intermediate layers of a neural network contain rich information for classification. For example, Lee et. al. [11] explored that the audio classification performance can be improved by concatenating features from different intermediate neural network layers. Features from multiple intermediate layers have been found to be effective not only for audio tasks, but also for computer vision tasks. For example, Meng et al. [12] proposed to extract features from different layers of a deep CNN and concatenated them to a representation which significantly outperforms the non-concatenated features [12].

Inspired by the success of multi-level representation [11, 12], we expand the single-level attention model [10] to a *multi-level attention model*. Multiple attention modules are applied on the intermediate neural network layers. Then, the outputs of the attention modules are concatenated to a vector followed by a fully connected layer with sigmoid non-linearity to predict the presence probability of each class.

The paper is organized as follows. Section II introduces related works. Section III introduces the single-level attention model [10]. Section IV describes the proposed multi-level attention module. Section V shows the experimental results. Section VI concludes and forecasts the future work.

2. RELATED WORKS

Audio classification: Audio classification has attracted many attention in recent years. Some representative challenges including DCASE 2013 [6], DCASE 2016 [2] and DCASE 2017 [1]. Hidden Markov models have been used to model audio events in [14]. Non negative matrix based methods were applied to learn the dictionary of audio events [15]. Recently, neural network based methods including fully connected neural networks [16], convolutional neural networks (CNN) [17] have been applied on audio classification and achieved the state-of-the-art performance.

Attention module: The concept of attention module is first introduced in natural language processing [18]. Attention module allows

deep neural networks to focus on relevant instances and ignore irrelevant instances in a bag. It has been successfully applied in machine translation [18], face detection [19], image classification [20] and captioning [21]. It is also utilized in the domain audio classification [22].

3. DATASET

AudioSet [7] consists of over two million samples. There are 527 classes in the current version. AudioSet is a multi-label dataset and each audio clip has one or several labels. Google created AudioSet through transfer learning. In the pre-training stage, two billion 10-second audio clips from YouTube covering more than 30,000 classes are collected and called YouTube 100M [23]. Log Mel spectrogram with size of 96×64 along time and frequency axis is extracted as feature for each audio clip. Then, a ResNet-50 model is trained using this YouTube 100M data. This trained ResNet-50 is later used as a feature extractor. After the pre-training stage, two million 10-second audio clips covering 527 classes are collected. The log Mel spectrogram of each audio clip is presented to the trained ResNet-50 model to extract the bottleneck features. In this process, each audio clip is compressed into 10 bottleneck features. Each feature has a dimension of 128. These two million samples constitute AudioSet.

4. SINGLE-LEVEL ATTENTION MODEL

In this section, we will introduce the single-level attention model proposed in [10].

To illustrate the notation, let x_t, $t = 1, 2, ..., T$ be the t-th bottleneck feature with a dimension $M = 128$. Each sample in AudioSet has $T = 10$ bottleneck features. $K = 527$ is the number of classes.

In the single-level attention model, each bottleneck feature x_t is presented to a trainable embedding mapping $f_{emb}(\cdot)$ to extract an embedded feature h_t:

$$h_t = f_{emb}(x_t) \tag{1}$$

Furthermore, an attention module is applied on the T embedded features to attain the class probabilities for the input sample:

$$y(\mathbf{h}) = \frac{1}{\sum_{t=1}^{T} v(h_t)} \sum_{t=1}^{T} v(h_t) f(h_t) \tag{2}$$

where $\mathbf{h} = [h_1, ..., h_T]$ is the concatenation of the embedded features. Non-negative function $v(\cdot)$ determines how much an embedded feature h_t should be attended or ignored and $f(\cdot)$ denotes the classification output on an embedded feature h_t. The attention module has ability to ignore irrelevant sound segments such as background noise and silences, and attend to the sound segments with audio events.

The implementation of the single-level attention model is shown in Fig. 1. The first part is an embedded mapping $f_{emb}(\cdot)$ modeled by three fully connected neural layers with H units. The second part is an attention module described by Equation (2). The attention non-negative mapping $v_k(\cdot)$ and the classification mapping $f_k(\cdot)$ are modeled by a softmax function and sigmoid function, respectively. The normalization applied after $v_k(\cdot)$ ensures the attention is normalized. Finally, the prediction is obtained by element-wise multiplication of the classification output and normalized attention output.

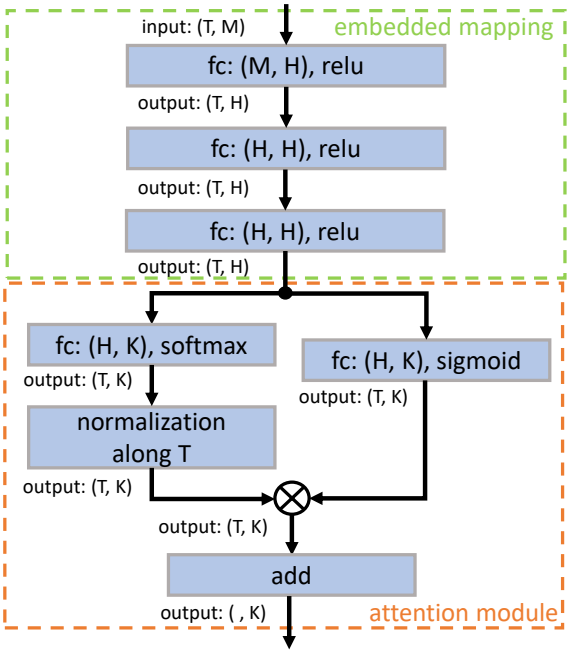


Figure 1: Architecture of the single-level attention model [10]

5. MULTI-LEVEL ATTENTION MODEL

Many works have explored that using multi-level features from intermediate layers of neural networks can promote the audio or image classification performance [11, 12]. We propose to extend the single-level attention model in Section 4 to multi-level attention model in our paper.

The architecture of the proposed multi-level attention model is shown in Fig. 2. Instead of applying a single-level attention model after the fully connected neural network, multiple attention modules are applied after intermediate layers as well. These attention modules aim to capture different level information. We denote the feedforward mappings as $g_l(\cdot)$ and the activations of the intermediate layers as $h^{(l)}$, where l is the number of embedded mappings. The feed-forward neural network can be written as:

$$\begin{cases} h_t^{(1)} = g_1(x_t) \\ h_t^{(l)} = g_l(h_t^{(l-1)}) \quad l = 2, 3, ..., L \end{cases} \tag{3}$$

where each forward mapping $g_l(\cdot)$ may consists of several fully connected layers in series (Fig. 2). For the single-level attention model, the prediction is produced by $y^{(L)} = y(\mathbf{h}^L)$ follows Equation (2) where $\mathbf{h}^{(L)} = \left[h_1^{(L)}, ..., h_T^{(L)}\right]$.

In the proposed multi-level attention model, each l-th attention module produces a prediction $y^{(l)} = y(\mathbf{h}^{(l)})$. Each prediction $y^{(l)} \in [0, 1]^K$. Then, all the predictions are concatenated to a vector $u \in [0, 1]^{KL}$:

$$u = \left[y^{(1)}, ..., y^{(L)}\right] \tag{4}$$

Finally, a fully connected layer followed by sigmoid nonlinearity is applied on the concatenated vector u to attain the class probabilities $z \in [0, 1]^K$ of the audio classes.

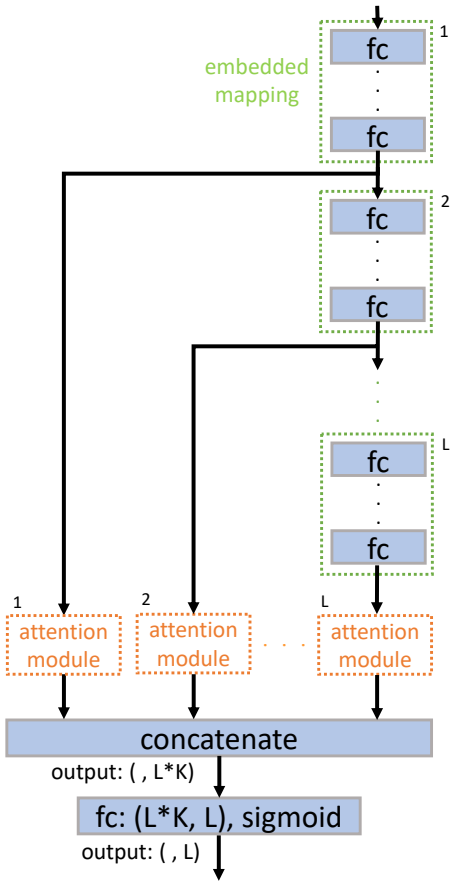


Figure 2: Architecture of the multi-level attention model

$$z = \phi(Wu + b) \tag{5}$$

where the $W \in R^{KL \times K}$ and $b \in R^K$ represent the weight matrix and the bias, separately. Sigmoid non-linearity $\phi(\cdot)$ is used for multi-label classification.

6. EXPERIMENTS

6.1. Training details

We use the balanced together with the unbalanced data from AudioSet [7] for training. We validated our model on the left out evaluation data of AudioSet. In order to comprehensively compare the performance of single-level and multi-level attention models, we implemented nine variants of single- (3-A, 6-A, 9-A) or multi-level attention models (1-A-1-A-1-A, 2-A-1-A, 2-A-2-A-2-A, 3-A-3-A, 3-A-3-A-3-A, 5-A-4-A) which are shown in Table I. The model 2-A-1-A represents two attention modules are applied after the 2nd and 3rd fully connected layers. The model 2-A-2-A-2-A represents three attention modules are applied after the 2nd, 4th and 6th fully connected layers. Each fully connected layer in all embedded mappings consists of 600 hidden units followed by ReLU activation function [24]. Dropout is used to prevent overfitting [25] with dropout rate of 0.4. Batch normalization [26] is applied to speed up training and prevent overfitting. We used Keras version 2.0.8 to implement our system. Adam optimizer [27] with learning rate of 0.001 is used. Batch size is set to 500. The setting of these hyper-parameters follows the configuration in [10]. Code has been made publicly available here [1]

6.2. Evaluation Metrics

To evaluate our model, we use three metrics of the Google's benchmark: mean average precision (mAP), area under curve (AUC) and d-prime. The mAP is the mean of average precision over all classes. The mAP is calculated by:

$$mAP = \frac{1}{K} \sum_{c=1}^{K} \sum_{n=1}^{N} p_{c,n} \Delta r_{c,n}, \tag{6}$$

where $p_{c,n}$ is the precision at n-th positive sample of c-th class. N is the number of positive samples for each class. $\Delta r_{c,n}$ is equal to $\frac{1}{N}$.

The AUC is area under the true positive-false positive rate curve. True positive rate (TPR) is a probability of correctly classifying a positive sample. False negative rate (FPR) is a probability of incorrectly classifying a negative sample as positive.

The d-prime is a deterministic function of AUC used in [7]. The d-prime can be calculated from AUC:

$$d\text{-}prime = \sqrt{2} F_x^{-1}(AUC) \tag{7}$$

F_x^{-1} is inverse of the cumulative distribution function and defined by:

$$F_x(x) = \int_{-\infty}^{x} \frac{1}{\sqrt{2\pi}} e^{\frac{-(x-\mu)^2}{2}} dx \tag{8}$$

The larger AUC and d-prime indicates the better the audio classification performance.

6.3. Analysis

The first two rows in Table I show the results of Google's benchmark [7] without attention model and Kong's result with the single-level attention model [10]. All of the multi-level attention models outperform Google's baseline and single-level attention model in mAP, AUC, and d-prime. The best multi-level attention model is 2-A-1-A with two attention modules on the 2nd and 3rd intermediate layers. A mAP of 0.360 is achieved, outperforming the single-level attention model [10] of 0.327 and the Google's baseline system of 0.314 [7]. The reason for the good performance using multi-level attention model is that the multi-level features extracted from the intermediate layers provide various representations, and then each attention module can filter the unrelated information of each feature. In addition, different classes may favor different layer of features and the last fully connected layer of each multi-level attention model can automatically select best feature for each class by the weight parameters.

When comparing all variants of the single-level attention model (3-A, 6-A, 9-A), it was observed that the performance notably degrades as the number of fully connected layers is increased. This results from that the features extracted from a deep fully connected layer (e.g. 6th and 9th fully connected layer) are worse than that of a shallow layer (e.g. 3rd fully connected layer).

[1] https://github.com/ChangsongYu/Eusipco2018_Google_AudioSet

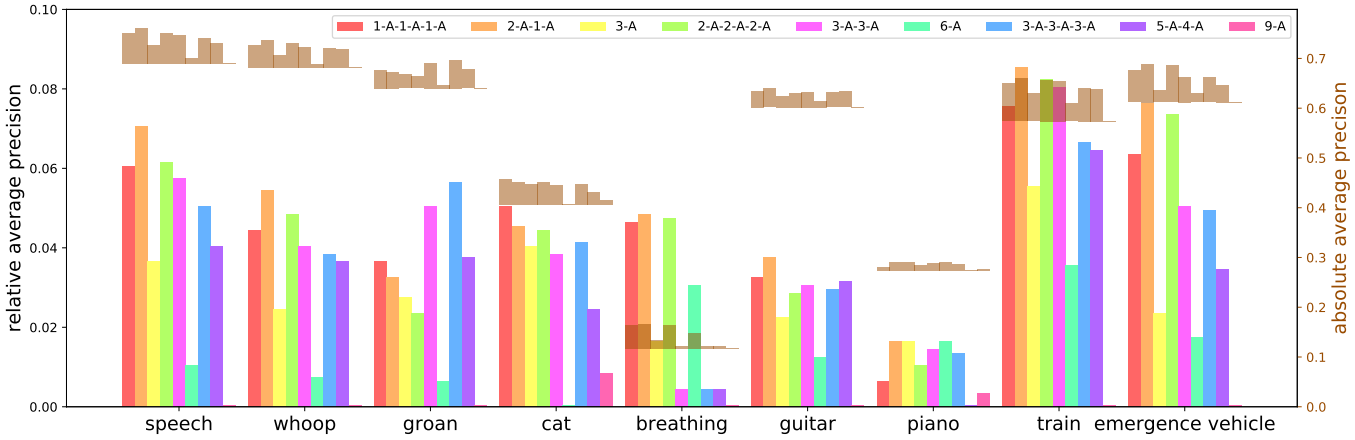


Figure 3: Average precision (AP) results of all single-level or multi-level attention models for nine randomly selected classes. The left black bar-graph scaled by the y-axis on the left-side represents the relative AP to the lowest AP among all models on a class. For example, the lowest AP among all models of the class "speech" is the AP of 9-A. The relative AP of 9-A of this class is 0 and that of 5-A-4-A is 0.04. The right brown bar-graph scaled by the y-axis on the right side represents the absolute AP. For example, the APs of 5-A-4-A and 9-A for the class "speech" are 0.730 and 0.690, separately.

Table 1: Comparisons of results of multi-level attention model

Model	**mAP**	**AUC**	**d-prime**
Benchmark	0.314	0.9590	2.452
Kong [10]	0.327	0.9650	2.558
1-A-1-A-1-A	0.357	0.9693	2.645
2-A-1-A	**0.360**	**0.9700**	**2.660**
3-A	0.336	0.9668	2.596
2-A-2-A-2-A	0.358	0.9695	2.650
3-A-3-A	0.355	0.9690	2.639
6-A	0.311	0.9571	2.430
3-A-3-A-3-A	0.353	0.9687	2.633
5-A-4-A	0.340	0.9676	2.612
9-A	0.305	0.9388	2.185

6.4. Performance visualization of individual classes

In addition, we investigate all variants of our single-level or multi-level attention model by comparing average precision (AP) of nine randomly selected classes are shown in Figure 3. For each class, the color bars plotted below is the relative improvement of AP and the bars plotted above is the absolute AP. The APs of classes such as speech and whoop are close to 0.7. In contrast, APs of many classes such as breathing are lower than 0.2.

Figure 3 shows that the multi-level attention models do not always achieve better performance on all classes than the single-level attention models. For the class "piano", the model 6-A outperforms the models 2-A-2-A-2-A and 3-A-3-A. We also observe that different classes favor different models. For example, the classes “speech”, “whoop”, “breathing”, “guitar”, “train” and “emergence vehicle” favor the model 2-A-1-A. However, the class "groan" favors the model 3-A-3-A-3-A. Overall, we can ensure that the performance of classification consistently increases on most classes when the multi-level features are concatenated and 2-A-1-A is the best architecture.

7. CONCLUSION

In this work, we introduced a multi-level attention model in addressing weakly labelled audio classification problem on AudioSet. The experimental results showed the effectiveness of multi-level attention models and achieved a state-of-the-art mean average precision (mAP) of 0.360 than the single-attention model and Google’s baseline system. In future, we will investigate the combination of the multi-scale and multi-level features for AudioSet classification.

8. ACKNOWLEDGEMENT

Qiuqiang Kong was supported by EPSRC grant EP/N014111/1 “Making Sense of Sounds” and a Research Scholarship from the China Scholarship Council (CSC) No. 201406150082.

9. REFERENCES

[1] A. Mesaros, T. Heittola, A. Diment, B. Elizalde, A. Shah, E. Vincent, B. Raj, and T. Virtanen, “DCASE 2017 challenge setup: Tasks, datasets and baseline system,” in *DCASE 2017-Workshop on Detection and Classification of Acoustic Scenes and Events*, 2017.

[2] A. Mesaros, T. Heittola, and T. Virtanen, “TUT database for acoustic scene classification and sound event detection,” in *Signal Processing Conference (EUSIPCO), 2016 24th European*. IEEE, 2016, pp. 1128–1132.

[3] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “Imagenet: A large-scale hierarchical image database,” in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE, 2009, pp. 248–255.

[4] J. Salamon, C. Jacoby, and J. P. Bello, “A dataset and taxonomy for urban sound research,” in *Proceedings of the 22nd*

ACM international conference on Multimedia. ACM, 2014, pp. 1041–1044.

[5] K. J. Piczak, "ESC: Dataset for environmental sound classification," in *Proceedings of the 23rd ACM international conference on Multimedia*. ACM, 2015, pp. 1015–1018.

[6] D. Stowell, D. Giannoulis, E. Benetos, M. Lagrange, and M. D. Plumbley, "Detection and classification of acoustic scenes and events," *IEEE Transactions on Multimedia*, vol. 17, no. 10, pp. 1733–1746, 2015.

[7] J. F. Gemmeke, D. P. Ellis, D. Freedman, A. Jansen, W. Lawrence, R. C. Moore, M. Plakal, and M. Ritter, "Audio set: An ontology and human-labeled dataset for audio events," in *IEEE ICASSP*, 2017.

[8] O. Maron and T. Lozano-Pérez, "A framework for multiple-instance learning," in *Advances in neural information processing systems*, 1998, pp. 570–576.

[9] A. Kumar and B. Raj, "Audio event detection using weakly labeled data," in *Proceedings of the 2016 ACM on Multimedia Conference*. ACM, 2016, pp. 1038–1047.

[10] Q. Kong, Y. Xu, W. Wang, and M. D. Plumbley, "Audio set classification with attention model: A probabilistic perspective," *arXiv preprint arXiv:1711.00927*, 2017.

[11] J. Lee and J. Nam, "Multi-level and multi-scale feature aggregation using pretrained convolutional neural networks for music auto-tagging," *IEEE signal processing letters*, vol. 24, no. 8, pp. 1208–1212, 2017.

[12] X. Meng, B. Leng, and G. Song, "A multi-level weighted representation for person re-identification," in *International Conference on Artificial Neural Networks*. Springer, 2017, pp. 80–88.

[13] H. R. Roth, L. Lu, A. Farag, H.-C. Shin, J. Liu, E. B. Turkbey, and R. M. Summers, "Deeporgan: Multi-level deep convolutional networks for automated pancreas segmentation," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, 2015, pp. 556–564.

[14] Y.-T. Peng, C.-Y. Lin, M.-T. Sun, and K.-C. Tsai, "Healthcare audio event classification using hidden markov models and hierarchical hidden markov models," in *Multimedia and Expo, 2009. ICME 2009. IEEE International Conference on*. IEEE, 2009, pp. 1218–1221.

[15] V. Bisot, R. Serizel, S. Essid, and G. Richard, "Supervised nonnegative matrix factorization for acoustic scene classification," *IEEE AASP Challenge on Detection and Classification of Acoustic Scenes and Events (DCASE)*, 2016.

[16] Q. Kong, I. Sobieraj, W. Wang, and M. Plumbley, "Deep neural network baseline for dcase challenge 2016," *Proceedings of DCASE 2016*, 2016.

[17] K. Choi, G. Fazekas, and M. Sandler, "Automatic tagging using deep convolutional neural networks," *arXiv preprint arXiv:1606.00298*, 2016.

[18] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[19] S. Sharma, R. Kiros, and R. Salakhutdinov, "Action recognition using visual attention," *arXiv preprint arXiv:1511.04119*, 2015.

[20] K. J. Shih, S. Singh, and D. Hoiem, "Where to look: Focus regions for visual question answering," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 4613–4621.

[21] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio, "Show, attend and tell: Neural image caption generation with visual attention," in *International Conference on Machine Learning*, 2015, pp. 2048–2057.

[22] Y. Xu, Q. Kong, W. Wang, and M. D. Plumbley, "Large-scale weakly supervised audio classification using gated convolutional neural network," *arXiv preprint arXiv:1710.00343*, 2017.

[23] S. Hershey, S. Chaudhuri, D. P. Ellis, J. F. Gemmeke, A. Jansen, R. C. Moore, M. Plakal, D. Platt, R. A. Saurous, B. Seybold *et al.*, "Cnn architectures for large-scale audio classification," in *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*. IEEE, 2017, pp. 131–135.

[24] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th international conference on machine learning (ICML-10)*, 2010, pp. 807–814.

[25] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[26] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*, 2015, pp. 448–456.

[27] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

META LEARNING BASED AUDIO TAGGING

Kele Xu[1]*, Boqing Zhu*[1]*, Dezhi Wang*[2]*, Yuxing Peng*[1]*, Huaimin Wang*[1]*, Lilun Zhang*[2]*, Bo Li*[3]

[1] National University of Defense Technology, Computer Dept., Changsha, China,
kelele.xu@gmail.com, zhuboqing09@nudt.edu.cn, pengyuxing@aliyun.com, whm_w@163.com
[2] National University of Defense Technology, College of Meteorology and Oceanography, Changsha, China,
wang_dezhi@hotmail.com, zll0434@163.com
[3] Beijing University of Posts and Telecommunications, Automation Dept., Beijing, 100876, China,
deepblue.lb@gmail.com

ABSTRACT

In this paper, we describe our solution for the general-purpose audio tagging task, which belongs to one of the subtasks in the DCASE 2018 challenge. For the solution, we employed both deep learning methods and statistic features-based shallow architecture learners. For single model, different deep convolutional neural network architectures are tested with different kinds of input, which ranges from the raw-signal, log-scaled Mel-spectrograms (log Mel) to Mel Frequency Cepstral Coefficients (MFCC). For log Mel and MFCC, the delta and delta-delta information are also used to formulate three-channel features, while mixup is used for the data augmentation.

Using ResNeXt, our best single convolutional neural network architecture provides a mAP@3 of 0.967 on the public Kaggle leaderboard, 0.939 on the private leaderboard. Moreover, to improve the accuracy further, we also propose a meta learning-based ensemble method. By employing the diversities between different architectures, the meta learning-based model can provide higher prediction accuracy and robustness with comparison to the single model. Our solution achieves a mAP@3 of 0.977 on the public leaderboard and 0.951 as our best on the private leaderboard, while the baseline gives a mAP@3 of 0.704.

Index Terms— Audio tagging, convolutional neural networks, meta-learning, mixup

1. INTRODUCTION

With the increase of smart mobile devices in recent years, huge amounts of user generated sound recordings are uploaded to the web every day [1]. Thus, the demand for analyzing these audio signals is increasing dramatically, for example, audio scene classification [2], automatic audio tagging [3]. Indeed, audio tagging task, the problem of predicting the presence or absence of certain acoustic events in the acoustic scenes, has drawn lots of attention during the last several years, due to its widely applications.

Historically, audio tagging has been addressed with different handcrafted features and shallow-architecture classifiers. The classifiers include: GMMs, HMMs, NMF and SVMs [4, 5, 6, 7]. Since the developments are rapid in signal processing and machine learning domains, there is an increasing interest in applying deep learning approaches for the audio tagging task [8, 9, 10, 11]. However, it remains challenging and falls short of accuracy and efficiency, and no reliable automatic general-purpose audio tagging systems exists. We argue that several factors lead to the phenomenon:

(1). Due to the lack of large-scale labeled data, the progress of audio tagging is far behind the analogous problem in the computer vision field. The audio-based approaches have been under-explored, and the state-of-the-art audio-based techniques are not able to achieve the comparable performance to its image/video counterpart. In fact, audios can sometimes be more descriptive than videos/images, especially when it comes to the description of an event.

(2). For the sound data, the number of sound events is huge, and the data quality is also of great diversity. Thus, it is important to handle the noisy training data, as the reliability of annotations is varying.

(3). Both shallow-architecture classifier using handcrafted features and deep learning approach should be employed together, which is under-explored. As demonstrated in many previous studies, efficient fusion between different models can boost the performance dramatically.

In this paper, we aim to build an general-purpose audio tagging system that can categorize an audio clip [12] as belonging to one of a set of 41 categories drawn from the AudioSet Ontology [13] (e.g., applause, bark, bus, animals, etc.). In more detail, our system has two levels: single deep model in the first level and the meta-learning in the second level. For single model in the first level, only convolutional neural networks are investigated, and different network architectures are tested with different kinds of input, which ranges from the raw-signal, log-scaled Mel-spectrograms (log Mel) to Mel Frequency Cepstral Coefficients (MFCC). For log Mel and MFCC, the delta and delta-delta information are also used to formulate three-channel features. Inception, ResNet, ResNeXt, Dual Path Networks (DPNs) are selected as the neural network architectures, while mixup is used for the data augmentation.

For the second level, to improve the classification further, we explore the use of meta-learning based method for the component classifier ensemble. Moreover, we propose to add the hand-crafted statistic features into the second level. In our experiments, this kind of ensemble method can provide superior accuracy and robustness. The paper is organized as follows. Section 2 gives the data augmentation method, while the brief introduction of the employed single models is presented in section 3. Section 4 describes the proposed meta-learning method and the brief experimental results.

2. DATA AUGMENTATION

The disadvantage of small dataset is that the model tends to overfitting. Currently, most publicly available audio tagging datasets have limited sizes [14]. To overcome this problem, we randomly extract a snippet of the original audio signal with equal length, 1.5 seconds. In this paper, we explore the use of mixup data augmentation [15]. In more detail, virtual training examples can be constructed by using the following formula:

$$x = \alpha \times x_i + (1 - \alpha) \times x_j \quad (1)$$

$$y = \alpha \times y_i + (1 - \alpha) \times y_j \quad (2)$$

where $(x_i,\ y_i)$ and $(x_j,\ y_j)$ are two examples randomly selected from the training batch. α is the mixed ratio. In our experiments, $\alpha \in Beta(3, 3)$. It is worthwhile to notice that the training samples can be either the raw wave signal segments or the time-frequency representations of the signal segment.

3. SINGLE MODEL

We used three kinds of inputs to train the network: raw wave signal, log-mel of the audio segment, and MFCC of the audio segment. In the papers [16, 17], we observed the complementarity of different features, so different features are used to improve performance. We select a 1.5s section randomly from the audio and input it into the network. The selected section is different in each epoch. When we take raw wave as input. We directly input $1.5 \times 44100 = 66150$ samples. When we take log-mel or MFCC as input, we extract a 64-dimensional log-mel and MFCC feature with a frame width of 80ms and a frame shift of 10ms, then we calculate the delta and delta-delta features of log-mel and MFCC with a window size of 9. Then we concatenate log-mel or MFCC with delta and delta-delta features to form a $3 \times 64 \times 150$ dimension input [18]. Two different ways are used to train the model: using ImageNet-based pre-trained model to initialize the weights, and training the weight from scratch. For the neural network architectures, 6 different model architectures are used.

3.1. Xception

Xception [19] is a deep convolutional neural network architecture inspired by Inception, where Inception modules have been replaced with depth-wise separable convolutions. In our experiments, Inception-V3 is employed with the log mel as input.

3.2. ResNet

ResNet makes the network deeper through a residual learning [20]. Instead of expecting each few stacked layers directly fit a desired underlying mapping, ResNet explicitly let these layers fit a residual mapping. Formally, denoting the desired underlying mapping as $H(x)$, the stacked nonlinear layers fit another mapping of $F(x) := H(x) - x$. The original mapping is recast into $F(x) + x$.

3.3. ResNeXt

ResNeXt [21] is a successful improvement based on ResNet. In more detail, ResNeXt is constructed by repeating a building block that aggregates a set of transformations with the same topology. Experiments on image classification demonstrate that increasing cardinality is a more effective way of gaining accuracy than going deeper or wider, especially when depth and width starts to give diminishing returns for existing models. The cardinality and the width of bottleneck are chosen as 32 and 4 respectively.

3.4. SE-ResNeXt

By introducing a new architectural unit, which we term the Squeeze-and-Excitation (SE) block [22], networks could improve the representational power by explicitly modeling the interdependencies between the channels of its convolutional features. The SE block takes into account another relationship besides spatial relations: the channel relationship. It allows the network to perform feature recalibration, through which it can learn to use global information to selectively emphasize informative features and suppress less useful ones. We apply the SE block on the ResNeXt, which is denoted as SE-ResNeXt in Table 1.

3.5. Wave-ResNeXt

To process the raw wave-form, we use a one-dimensional convolution to simulate a band-pass filter to extract features. Moreover, In order to obtain more complementary features, we use multi-scale convolution to the original signal. Just like the multi-scale feature extraction process we designed in [17], the backend network is replaced by the ResNeXt.

3.6. DPN

Residual Network (ResNet) enables feature re-usage while Densely Convolutional Network (DenseNet) enables new features exploration which are both important for learning good representations. To enjoy the benefits from both path topologies, Dual Path Network (DPN) [23] shares common features while maintaining the flexibility to explore new features through dual path architectures.

The detail performances of different single models are given in Table 1. The number following the network name represents the number of layers in the network, for example ResNet50 means ResNet with the configuration of 50 layers. As can be seen from Table 1, we found ImageNet-based pre-traiend model can improve the performances, but the generalization error between the train and test data is increased. Further, the Log-Mel feature can achieve better performance than waveform or MFCC generally. Moreover, it is worthwhile to notice that mixup can boost the performance without any exception.

4. META LEARNING-BASED ENSEMBLE AND EXPERIMENTAL RESULTS

It is widely known that ensemble diverse classifiers can improve the accuracy and robustness for the classification task. However, the ensemble learning has been under-explored for the audio tagging task. Previous efforts employ linear regression for the ensemble learning. Here, unlike previous attempts, we explore the use of stacked generalization in multiple levels to improve accuracy and robustness in this multi-class classification problem. The framework is computational, scalable and it have been tested on multiple machine learning tasks. Fig. 1 shows the proposed stacking architecture used in our task, which is composed of two levels. We randomly split the data into 5 folds in our experiments. For each CNN, we run 5 individual CNN models for each fold, and one model to predict the probabilities for each sample in the validating set by using the whole training

Table 1: Performance comparison between different single models

Network architecture	Pretraiend	Input	Data augmentation	Public mAP@3	Private mAP@3
Wave-ResNext	Yes	wave	-	0.938	0.918
Wave-ResNext	No	wave	-	0.910	0.902
Xception	Yes	log mel	mixup	0.917	0.906
ResNet50	Yes	MFCC	mixup	0.932	0.914
ResNet50	Yes	log mel	mixup	0.950	0.932
ResNeXt101	Yes	log mel	-	0.935	0.924
ResNeXt101	Yes	log mel	mixup	0.967	0.939
ResNeXt101	No	log mel	mixup	0.921	0.887
SE-ResNeXt101	Yes	log mel	-	0.939	0.920
SE-ResNeXt101	Yes	log mel	mixup	0.950	0.927
DPN68	Yes	log mel	-	0.939	0.925
DPN68	Yes	log mel	mixup	0.950	0.922
DPN96	Yes	log mel	-	0.937	0.926
DPN96	Yes	log mel	mixup	0.964	0.936
DPN96	No	log mel	mixup	0.917	0.886
DPN107	Yes	log mel	mixup	0.957	0.938

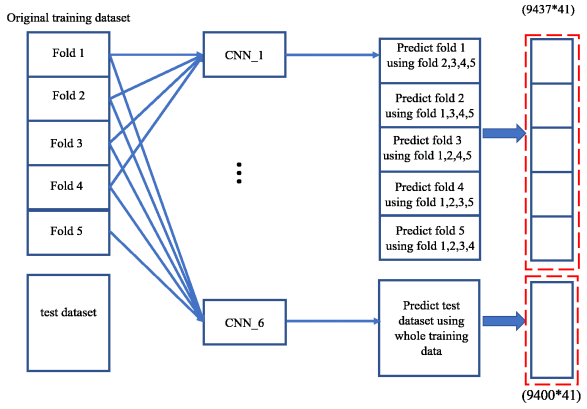


Figure 1: Meta-features construction using CNNs.

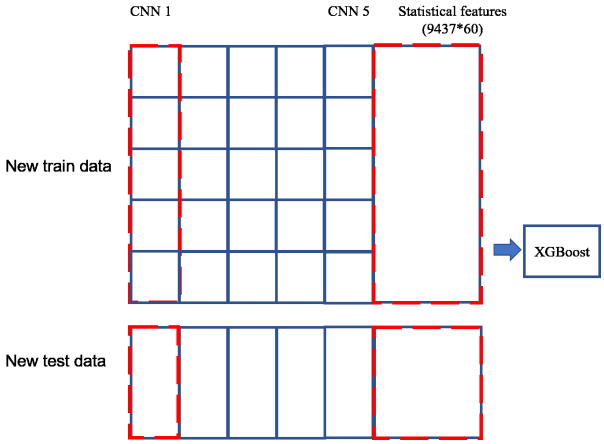


Figure 2: Meta-learning for audio tagging.

dataset. The predicted probabilities of different classes will be concatenated to generate meta-features. For each classifier, the probabilities for 41 classes will be used as the meta-features, which will be concatenated to generate the new training dataset (as can be seen in Fig.1), and the meta features will be used as the input for level 2.

In our experiments, the first layers are composed of 5 different CNN architectures: ResNeXt using the log mel with mixup, ResNeXt using the raw wave without mixup; ResNet using the log mel without mixup, ResNet using the wave with mixup, DPN using the log mel with mixup (as can be seen from Fig.2).

Except for the deep learning-based meta features, we also employ the traditional handcrafted features. In more detail, we calculate the max value, min value, variance value, skewness for the MFCC of the audio signal segment. And the statistical features are also used as the meta-features.

For level 2, we employ the widely used method gradient tree boosting machine for the multi-class classification task. The eXtreme Gradient Boosting method (XGBoost) [24, 25] library, a tree boosting machine based classification implementation, is selected as the benchmark. It is because that compared to other approaches (such as, linear regression, Support Vector Machine, Random Forest [26]), XGBoost provides better classification performance in our experiment, and its power has furthermore been validated on several public machine learning challenges. We use the default hyper parameters for the XGBoost, and the maximum depth is set to 3 to prevent overfitting.

Using the proposed meta-learning method, our solution achieves a mAP@3 of 0.977 on the public leaderboard and 0.951 as our best on the private leaderboard, while the baseline gives a mAP@3 of 0.70.

5. CONCLUSION

In this article, we proposed an effective meta-learning system employing both deep learning architectures and statistic features-based learners to achieve a successful solution for the general-purpose audio tagging task in DCASE 2018. A comparative study of the performance of several well-developed convolutional neural network architectures with different types of input were conducted to obtain excellent single models for the subsequent meta-learning. Mixup

technique was also implemented in the training process which constantly improves the model performance as expected. The XGBoost approach was applied on a hybrid combination of meta-features including deep-learning features and statistical features, which have a superb classification performance. The final results put us in the first place on the task public leaderboard with a mAP@3 of 0.977 and the fourth place on the private leaderboard. In future, we would like to further evaluate the performance of our method on the Google AudioSet.

6. REFERENCES

[1] E. Marchi, D. Tonelli, X. Xu, F. Ringeval, J. Deng, S. Squartini, and B. Schuller, "Pairwise decomposition with deep neural networks and multiscale kernel subspace learning for acoustic scene classification," in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2016 Workshop (DCASE2016)*, 2016, pp. 65–69.

[2] A. Mesaros, T. Heittola, and T. Virtanen, "A multi-device dataset for urban acoustic scene classification," 2018, submitted to DCASE2018 Workshop. [Online]. Available: https://arxiv.org/abs/1807.09840

[3] E. Fonseca, M. Plakal, F. Font, D. P. W. Ellis, X. Favory, J. Pons, and X. Serra, "General-purpose tagging of freesound audio with audioset labels: Task description, dataset, and baseline," 2018, submitted to DCASE2018 Workshop. [Online]. Available: https://arxiv.org/abs/1807.09902

[4] P. Dhanalakshmi, S. Palanivel, and V. Ramalingam, "Classification of audio signals using aann and gmm," *Applied soft computing*, vol. 11, no. 1, pp. 716–723, 2011.

[5] I. Karpov and D. Subramanian, "Hidden markov classification for musical genres," *Course Project*, 2002.

[6] P. Giannoulis, G. Potamianos, and P. Maragos, "On the joint use of nmf and classification for overlapping acoustic event detection," in *Multidisciplinary Digital Publishing Institute Proceedings*, vol. 2, no. 2, 2018, p. 90.

[7] K. J. Piczak, "Environmental sound classification with convolutional neural networks," in *Machine Learning for Signal Processing (MLSP), 2015 IEEE 25th International Workshop on*. IEEE, 2015, pp. 1–6.

[8] S. Hershey, S. Chaudhuri, D. P. Ellis, J. F. Gemmeke, A. Jansen, R. C. Moore, M. Plakal, D. Platt, R. A. Saurous, B. Seybold, *et al.*, "Cnn architectures for large-scale audio classification," in *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*. IEEE, 2017, pp. 131–135.

[9] E. Fonseca, R. Gong, and X. Serra, "A simple fusion of deep and shallow learning for acoustic scene classification," *arXiv preprint arXiv:1806.07506*, 2018.

[10] Y. Xu, Q. Huang, W. Wang, P. Foster, S. Sigtia, P. J. Jackson, and M. D. Plumbley, "Unsupervised feature learning based on deep models for environmental audio tagging," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 25, no. 6, pp. 1230–1241, 2017.

[11] T. Hayashi, S. Watanabe, T. Toda, T. Hori, J. Le Roux, K. Takeda, T. Hayashi, S. Watanabe, T. Toda, T. Hori, *et al.*, "Duration-controlled lstm for polyphonic sound event detection," *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, vol. 25, no. 11, pp. 2059–2070, 2017.

[12] F. Font, G. Roma, and X. Serra, "Freesound technical demo," in *Proceedings of the 21st ACM international conference on Multimedia*. ACM, 2013, pp. 411–412.

[13] J. F. Gemmeke, D. P. Ellis, D. Freedman, A. Jansen, W. Lawrence, R. C. Moore, M. Plakal, and M. Ritter, "Audio set: An ontology and human-labeled dataset for audio events," in *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*. IEEE, 2017, pp. 776–780.

[14] J. Salamon, C. Jacoby, and J. P. Bello, "A dataset and taxonomy for urban sound research," in *ACM International Conference on Multimedia*, 2014, pp. 1041–1044.

[15] K. Xu, D. Feng, H. Mi, B. Zhu, D. Wang, L. Zhang, H. Cai, and S. Liu, "Mixup-based acoustic scene classification using multi-channel convolutional neural network," *arXiv preprint arXiv:1805.07319*, 2018.

[16] Y. Tokozume and T. Harada, "Learning environmental sounds with end-to-end convolutional neural network," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2017, pp. 2721–2725.

[17] B. Zhu, C. Wang, F. Liu, J. Lei, Z. Lu, and Y. Peng, "Learning environmental sounds with multi-scale convolutional neural network," *arXiv preprint arXiv:1803.10219*, 2018.

[18] D. Feng, K. Xu, H. Mi, F. Liao, and Y. Zhou, "Sample dropout for audio scene classification using multi-scale dense connected convolutional neural network," *arXiv preprint arXiv:1806.04422*, 2018.

[19] J. Carreira, H. Madeira, and J. G. Silva, "Xception: A technique for the experimental evaluation of dependability in modern computers," *IEEE Transactions on Software Engineering*, vol. 24, no. 2, pp. 125–136, 1998.

[20] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[21] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*. IEEE, 2017, pp. 5987–5995.

[22] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," *arXiv preprint arXiv:1709.01507*, vol. 7, 2017.

[23] Y. Chen, J. Li, H. Xiao, X. Jin, S. Yan, and J. Feng, "Dual path networks," in *Advances in Neural Information Processing Systems*, 2017, pp. 4467–4475.

[24] T. Chen, T. He, M. Benesty, *et al.*, "Xgboost: extreme gradient boosting," *R package version 0.4-2*, pp. 1–4, 2015.

[25] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. ACM, 2016, pp. 785–794.

[26] A. Liaw, M. Wiener, *et al.*, "Classification and regression by randomforest," *R news*, vol. 2, no. 3, pp. 18–22, 2002.

AUDIO TAGGING SYSTEM USING DENSELY CONNECTED CONVOLUTIONAL NETWORKS

Il-Young Jeong and Hyungui Lim

Cochlear.ai, Seoul, Korea
{iyjeong, hglim}@cochlear.ai

ABSTRACT

In this paper, we describe the techniques and models applied to our submission for DCASE 2018 task 2: *General-purpose audio tagging of Freesound content with AudioSet labels*. We mainly focus on how to train deep learning models efficiently against strong augmentation and label noise. First, we conducted a single-block DenseNet architecture and multi-head softmax classifier for efficient learning with mixup augmentation. For the label noise, we applied the batch-wise loss masking to eliminate the loss of outliers in a mini-batch. We also tried an ensemble of various models, trained by using different sampling rate or audio representation.

Index Terms— Audio tagging, DenseNet, Mixup, Multi-head softmax, Batch-wise loss masking

1. INTRODUCTION

Audio tagging is a research area to find one or more labels from audio signals. It has been studied in various fields including music tagging [1], domestic audio tagging [2], and acoustic scene classification [3]. Similar to audio tagging, sound event detection (SED) is another area of research to provide additional information about temporal boundaries of sound events. Despite some differences such as the optimal size of audio input or integration process of predicted values, their approaches are generally similar [4, 5]. Both of them have recently adopted deep learning-based approaches such as convolutional neural networks (ConvNet) [1, 6] and convolutional recurrent neural networks (CRNN) [7, 8]. In *Detection and classification of acoustic scenes and events* (DCASE) 2017, ConvNet-based [4, 9, 10] and CRNN-based [5, 11, 12, 13] methods outperformed conventional machine learning methods such as hidden Markov model (HMM) [14], non-negative matrix factorization (NMF) [15], and support vector machine (SVM) [16].

As the complexity of the model increases, preventing overfitting caused by lack of audio data has become more critical. To this end, data augmentation methods such as time stretching, pitch shifting, background noise mixing [17] and mixup [18] have been proposed. More recently, large-scale datasets such as Audioset [19] and Freesound dataset (FSD) [20] have been released. While data shortages have shown some improvements, there still exist difficulties associated with ambiguous relationship between an audio and its labels such as imprecisely labeled audio or many possible interpretations of a single sound.

In DCASE 2018, among the various tasks, task 2: *General-purpose audio tagging of Freesound content with AudioSet labels* aims to recognize and tag sound events of diverse nature, including musical instruments, human sounds, domestic sounds, animals, etc. In total of 41 sound event categories are considered, and the audio samples are provided from FSD.

The framework presented in this paper is based on ConvNet, specifically a densely connected ConvNet (DenseNet) [21]. We designed our models to have a single-block architecture, in which, all layers from the very bottom to the top are connected densely. In addition, we applied several techniques including mixup augmentation, multi-head softmax and batch-wise loss masking, expecting robust performance and efficient learning against audio-label ambiguities. Trained models and their ensembles were examined by using a variety of data representations, including low-level transformations and sampling rates.

2. DATASET

Freesound Dataset Kaggle 2018 (FSDKaggle2018) was provided for the challenge [22]. It consists of 11,703 audio recording data, where 9,473 recordings are for training and 1,600 are for evaluation. Each data is labeled into one of 41 audio event categories such as acoustic guitar, bus or laughter. All data is provided in a single-channel format with a sampling rate of 44.1kHz, while the duration is varied form 300 milliseconds to 30 seconds. The number of data per category is not balanced from 94 to 300.

One of the important features of FSDKaggle2018 is that only 3,710 training data labels were verified manually. In case of 5,763 recordings with the non-verified label, some may consist sounds belonging to other categories, or not belonging to any of 41 categories.

3. PROPOSED FRAMEWORK

3.1. Preprocessing and batch generation

Except for data resampling, the proposed framework does not have a preprocessing step. We tried several other techniques, including silence removal and pre-emphasis filtering, but we have not found any meaningful improvements. We applied 16kHz, 32kHz and 44.1kHz (original data) for data resampling. Low sampling rates may lose useful information at high frequencies, but its smaller data size allows to analyze longer time ranges with less computation.

We designed the batch generation framework as follows. First, we set each batch to have the same number of classes. In this work, one batch had one recordings for each class, thus the batch size was 41. We expected that it helps to make the optimization process to be stable and fast.

For efficient mini-batch learning, the length of input data in a mini-batch have to be fixed. We set it to be 64,000 samples, which is the same as 4s for 16kHz data and shorter for data with higher sampling rate. If the original recording is longer than this, a 64,000 sample segment was extracted at random offset. If the length is shorter, zero padding was applied to the beginning and end of the data.

3.1.1. Mixup augmentation

Mixup is an augmentation method which mixes two training data linearly [18]. Let x_i and t_i are i-th raw input data and corresponding binary labels in training dataset, respectively, then mixup generates an augmented data $\hat{x}$ which is a mixture of the two original data as follows:

$$\hat{x} = \lambda x_j + (1 - \lambda)x_k, \quad (1)$$

where $\lambda \in (0, 1)$. Similarly, the label of the generated data is set to be $\hat{t} = \lambda t_j + (1 - \lambda)t_k$. Despite its simplicity, mixup has shown meaningful improvements in image classification tasks.

We believe that mixup technique is also, or more, suitable for audio analysis, since the captured audio signal in real-world can be considered as a linear mixture of various 'source' signals. In this perspective, classifying $\hat{x}$ to $\hat{t}$ could be thought of as a task which detects multiple simultaneous sound events.

In this study, we set λ to be random variable of Beta distribution of $\alpha = \beta = 0.4$. In addition, we set $\lambda > 0.5$, so the data of target class, which is evenly distributed in a batch, is always predominant in the generated data. Another data class for mixup was randomly selected. Finally, we also applied the scale augmentation, which randomly scales the data. This process can be represented by the following equation.

$$\hat{x} = w\lambda x_j / \max(|x_j|) + w(1 - \lambda)x_k / \max(|x_k|), \quad (2)$$

where w is random variable with uniform distribution for the scale augmentation.

3.2. Model architecture

While mixup technique meaningfully prevents overfitting and increases validation/test accuracy, it also makes the minimization of training loss to be difficult. Therefore, our model and learning strategy are focused on efficient training against strong mixup augmentation.

The overall architecture of the presented model is presented in Fig. 1. And Fig. 2 shows the details of each module in the model. It is noted that we tried 2 different models, which are 'logmel-based' and 'waveform-based', while Fig. 2 only represents logmel-based model. The minor changes for waveform-based model are described in each subsection.

3.2.1. Low-level module

The logarithm of mel-scale spectrogram (logmel) has been widely used as a preprocessing step of audio analysis. In this work, we applied the logmel transform as a low-level module in our model and implemented it using kapre. [23].

Detailed low-level module is described in Fig. 2 (a). First, the input waveform of a size (64000, 1), which denotes (sample, feature), is normalized by using Batch normalization (BN) [24], then transformed into a logmel domain with two dimensions, time and frequency. For the logmel transformation, we used 1024 window size with 128 shift and 64 mel-frequency bins. After applying BN, considering the frequency bins as a filter, it is reshaped to a size (time, frequency, 1) and considered as a grayscale image. As described in the next subsection, we aimed to conduct a single-block densely-connected architecture, so the output features of convolution layer is concatenated with its inputs.

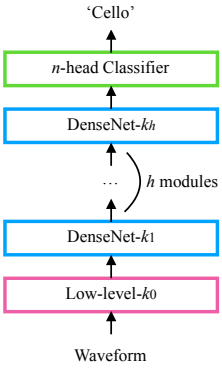


Figure 1: Overall architecture of the presented models. Details of each module are shown in Fig. 2

For the waveform-domain model, the low-level module is simplified and modified as follows:

- Logmel and BN+Reshape layers are removed and input data after BN is directly concatenated to Conv outputs.
- 3x3 Conv layer is replaced to 1x3 Conv.

3.2.2. DenseNet Module

We designed our model based on DenseNet [21]. Although the original DenseNet model divided its architecture into several blocks and applied densely-connected layer within each block, our model consists of a single block architecture so the very first logmel or waveform can be reached even to the very last layer. In our experiments, increasing the number of densely-connected blocks, or the number of layer that disconnects the concatenation, slows down the training speed.

Fig. 2 (b) shows the details of DenseNet module. Because the size of the filter continues to increase over concatenation, 1x1 convolution is first applied to reduce it before 3x3 convolution. We also applied Squeeze-and-Excitation Network [25] to support efficient training by adding a few parameters. 2x2 max-pooling is applied to the last layer of each DenseNet module.

For waveform-based model, it was modified as follows:

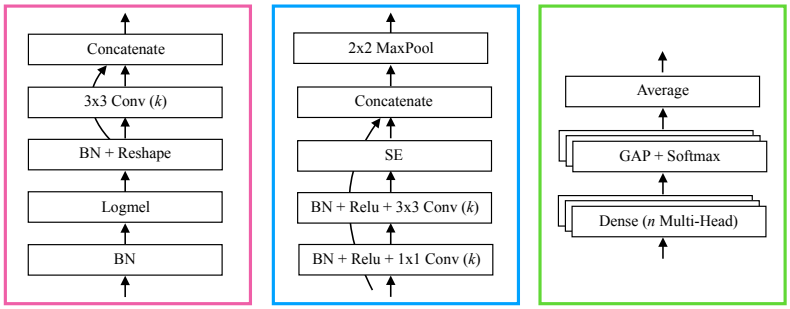
- 3x3 convolution is replaced by 1x3 convolution.
- 2x2 max-pooling is replaced by 1x2 max-pooling.

3.2.3. Classifier module

In general, the goal of classification task is to predict a binary target output vector such as [1, 0, 0]. When mixup is applied, on the other hand, it needs to predict the real values in the range of (0, 1) such as [0.9, 0.1, 0] or [0.7, 0.3, 0]. When a stronger mix-up is applied, the more target values tend to be close to 0.5.

To efficiently train the mixup model, we modified the existing softmax output layer to have a multi-head architecture, where output is obtained by averaging multiple softmax outputs as Fig. 2 (c).

We expect it will be helpful particularly in training with a strong mixup augmentation for the following reasons. Since the target values of augmented data are in the range of (0, 1), the values of each softmax can be varied even if these average is same as its target. Moreover, because softmax output is bounded in the range of (0,



(a) Low-level-*k* module (b) DenseNet-*k* module (c) *n*-head classifier module

Figure 2: Details of each module in the presented model. k and n denote the filter size of convolution and the number of softmax layer respectively. BN: batch normalization, Concatenate: feature concatenation, Relu: rectified linear unit, Conv: linear convolution, MaxPool: max-pooling, GAP: global average pooling.

1), more margin is allowed when the target is closed to 0.5. In experiments using various n-multi-head settings, we found that larger n helps to accelerates the training procedure, while its maximum validation accuracy did not show the meaningful difference.

3.2.4. Overall frameworks

Our entire model is conducted by using above mentioned modules. For the logmel- and waveform-based model, the detailed parameters for each module is as follows.

- Logmel-based: Low-level-15, 8 DenseNet modules of k=(16, 32, 64, 128, 256, 512, 512, 512), 8-head Classifier. About 11M trainable parameters.
- Waveform-based: Low-level-1, 15 DenseNet modules of k=(2, 4, 8, 16, 32, 64, 128, 256, 512, $\cdots$, 512), 8-head Classifier. About 16M trainable parameters.

3.3. Optimization

Our experiment was implemented using Keras [26]. Adam [27] was used for optimization. Although it adaptively controls the learning rate (*lr*) by itself, we found that manual decay of learning rate helps the optimization even for Adam. We set *lr* to be 10^{-3} for first 150k mini-batch iterations, 10^{-4} for next 100k and 10^{-5} for the last 50k. Note that the actual learning rate for each minibatch is based on this *lr* parameter and adaptation algorithm of Adam.

Validation accuracy was evaluated for every 1k minibatch iteration and the best model was saved for the evaluation. The computation time for 1k iteration was about 150 s (logmel-based model) and 200 s (waveform-based model) using NVIDIA Tesla P100 GPU.

3.3.1. Batch-wise loss masking

Another consideration for optimization was label noise. The 3.7k data was verified from the 9.5k data for training and validation and the remainder was not guaranteed the true label. In this case, this data with false labels may not only lead to lower classification performance, but also disturb optimization because the model is trained to handle those outliers. Therefore, we believed that it will be helpful if those noise data can be detected and eliminated.

In this work, we used an iterative detection strategy which is called batch-wise loss masking in this paper. First, the conventional loss function for a mini-batch is defined as

$$J = \sum_{n} C_n, \tag{3}$$

where C_n is cross-entropy for a single data in a mini-batch, which is defined as

$$C_n = -\sum_{c} t_{n,c} \log(y_{n,c}), \tag{4}$$

where $t_{n,c}$ and $y_{n,c}$ denote the label and classification results for c-th class of n-th data, respectively. On the other hand, if we know which data is labeled correctly and which is not, we can modify the loss function to ignore the noise data as follows:

$$\hat{J} = \sum_{n} m_n C_n, \tag{5}$$

where m_n is 1 if the n-th data is correctly labeled and 0 if not. Since the optimal m is not known in the real-world situations, it is required to be estimated.

In this study, we used two factors to determine the values of m. First, verified data can always be considered as a true label. On the other hand, if some data show particularly high loss in the current model, then it can be considered as an outlier with wrong label. From these factors, we set m for each minibatch iteration as folllows:

$$m_n = \begin{cases} 1 & \text{if } v_n = 1 \text{ or } C_n < \mu, \\ 0 & \text{otherwise,} \end{cases} \tag{6}$$

where v_n denotes whether n-th data is manually verified or not. μ is defined as follows in this work:

$$\mu = \alpha \times \max_{n} C_n, \tag{7}$$

where α was empirically set to be 0.8 in this work. This modification removes some data with the largest errors in the gradient calculation. In addition, since the outlier data is selected in a batch, it is expected that data with noise will be gradually found. In our experiments, this masking technique improved the cross-validation accuracy about 1 percent point.

3.4. Inference and ensemble

Unlike the training phase, the entire data longer than 64,000 sample is fed directly into the model. The presented model can handle the

Table 1: Comparison of MAP@3 scores for models using different audio representations and sampling rates.

Model	16kHz, 4s	32kHz, 2s	44.1kHz, 1.45s	ensemble
logmel	0.932	0.940	0.942	0.947
waveform	0.924	0.925	0.915	0.933
ensemble	0.944	0.949	0.948	**0.954**

variable length data and the output size is always the same due to the global average pooling layer. However, we applied zero padding for shorter data, because we believe that the length of zero-padded region implies the information about the data length, which can be an important clue for recognition.

For the ensemble of multiple models, we took the geometric mean of the model outputs. The logmel-based model was given a weight of 1.5, considering that it outperformed the waveform-based model in our experiments. The ensemble process can be represented as follows:

$$\tilde{y} = \exp(\sum_{e} (p_e \log y_e)), \tag{8}$$

where y_e and p_e denote the output and the ensemble weight of e-th model, respectively. The final output was obtained after normalizing $\tilde{y}$ to its l_1-norm.

4. RESULTS

4.1. Comparison of the low-level modules, sampling rate and temporal length

The first experiment evaluated classification performance at various low-level modules and sampling rates. It is noted that changing sampling rate directly affects to the temporal length of the analysis window since the number of samples in the batch generation was fixed.

Each model/sampling rate setting is conducted by using ensembles of 5 cross-validation models, and Table 1 shows those MAP@3[1] score. From those results, we found that logmel-based models outperform waveform-based ones, while those ensemble shows meaningful improvements. Although the results were less sensitive to sampling rate, however, it seems that the higher sampling rate leads to better classification performance, particularly in case of logmel-based models. Again, ensembles of models with various sampling rate/temporal length improves MAP@3 score. The ensemble of all different settings achieved 0.954, which is the state-of-the-art in this task[2].

4.2. Effects of multi-head classifier

The main aim of the multi-head classifier module was to accelerate the minimization of training loss. To observe the effect of the number of softmax layers, we compared the history of losses over mini-batch iteration. For this experiments, we used the logmel-based model and 44.1kHz sampling rate. Other settings were the same as previous experiments (mixup, batch-wise loss masking, learning rate, etc.). Fig. 3 shows the convergence of training loss for $n = 1$ and $n = 8$ in the first 100k iteration. Although the number of parameters in 7 added layers is around 0.6M, which is only 6% of $n = 1$ model, but the $n = 8$ model shows meaningful faster convergence.

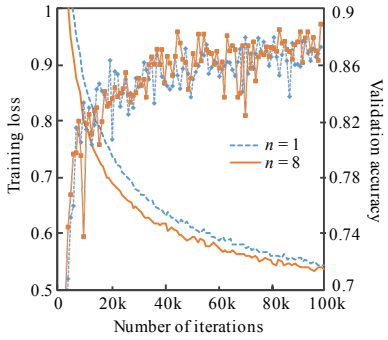


Figure 3: Training loss convergence and validation accuracies of 1-head and 8-head architecture. Loss over 1 and accuracy below 0.7 are clipped for visual convenience.

We leave the following discussions as future works. At first, the effect of multi-head layer on the test data have to be verified from more experiments. In addition, while the single-block DenseNet architecture has many filters in the last hidden layer, we expect that the number of parameters of 0.6M can be reduced by modifying the model architecture.

5. DISCUSSION

The proposed framework has shown meaningful results in the challenge, but there is room for improvement. First, the presented techniques, including single-block DenseNet architecture, Squeeze-and-Excitation Network, multi-head softmax and batch-wise loss masking, require further experimentation in various condition for verify its effectiveness. Here, the various condition may include applying to different models, datasets or tasks.

We also plan to improve the classification performance of waveform-based model. Although the logmel operation is similar to convolution layer except the squared and log operations, the waveform-based model showed relatively a lower performance compared to the logmel-based model. We believe that improving waveform-based model will be also helpful for the ensemble result.

Another important consideration is minimization of model size, which is currently 11M to 19M parameters for a single model. The smaller the size of model is, the easier to be implemented in devices with less power consumption and smaller size. Therefore, finding the minimal model size maintaining the detection performance will improve the usability in real-world applications.

6. CONCLUSION AND FUTURE WORKS

This paper described the audio tagging system submitted in DCASE 2018 task 2. We primarily focused on finding a technique that efficiently learns strongly augmented data. We presented a single-block DenseNet model, multi-head softmax layer, as well as batch-wise loss masking. We also tried to ensemble models of various low-level modules and sampling rate, and it achieved the state-of-the-art results.

[1] https://www.kaggle.com/c/freesound-audio-tagging#evaluation
[2] https://www.kaggle.com/c/freesound-audio-tagging/leaderboard

7. REFERENCES

[1] K. Choi, G. Fazekas, and M. Sandler, "Automatic tagging using deep convolutional neural networks," *arXiv preprint arXiv:1606.00298*, 2016.

[2] Y. Xu, Q. Huang, W. Wang, P. J. Jackson, and M. D. Plumbley, "Fully dnn-based multi-label regression for audio tagging," *Detection and Classification of Acoustic Scenes and Events (DCASE)*, 2016.

[3] S. Mun, S. Park, D. K. Han, and H. Ko, "Generative adversarial network based acoustic scene training set augmentation and selection using svm hyper-plane," *Detection and Classification of Acoustic Scenes and Events (DCASE)*, 2017.

[4] D. Lee, S. Lee, Y. Han, and K. Lee, "Ensemble of convolutional neural networks for weakly-supervised sound event detection using multiple scale input," *Detection and Classification of Acoustic Scenes and Events (DCASE)*, 2017.

[5] Y. Xu, Q. Kong, W. Wang, and M. D. Plumbley, "Surrey-cvssp system for dcase2017 challenge task4," *Detection and Classification of Acoustic Scenes and Events (DCASE)*, 2017.

[6] H. Zhang, I. McLoughlin, and Y. Song, "Robust sound event recognition using convolutional neural networks," in *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*. IEEE, 2015, pp. 559–563.

[7] E. Cakir, S. Adavanne, G. Parascandolo, K. Drossos, and T. Virtanen, "Convolutional recurrent neural networks for bird audio detection," in *Signal Processing Conference (EUSIPCO), 2017 25th European*. IEEE, 2017, pp. 1744–1748.

[8] Y. Hou, Q. Kong, and S. Li, "Audio tagging with connectionist temporal classification model using sequential labelled data," *arXiv preprint arXiv:1808.01935*, 2018.

[9] Y. Han, J. Park, and K. Lee, "Convolutional neural networks with binaural representations and background subtraction for acoustic scene classification," *the Detection and Classification of Acoustic Scenes and Events (DCASE)*, 2017.

[10] I.-Y. Jeong, S. Lee, Y. Han, and K. Lee, "Audio event detection using multiple-input convolutional neural network," *Detection and Classification of Acoustic Scenes and Events (DCASE)*, 2017.

[11] H. Lim, J. Park, K. Lee, and Y. Han, "Rare sound event detection using 1d convolutional recurrent neural networks," *Detection and Classification of Acoustic Scenes and Events (DCASE)*, 2017.

[12] E. Cakır and T. Virtanen, "Convolutional recurrent neural networks for rare sound event detection," *Detection and Classification of Acoustic Scenes and Events (DCASE)*, 2017.

[13] S. Adavanne and T. Virtanen, "A report on sound event detection with different binaural features," *Detection and Classification of Acoustic Scenes and Events (DCASE)*, 2017.

[14] A. Mesaros, T. Heittola, A. Eronen, and T. Virtanen, "Acoustic event detection in real life recordings," in *Signal Processing Conference, 2010 18th European*. IEEE, 2010, pp. 1267–1271.

[15] A. Mesaros, T. Heittola, O. Dikmen, and T. Virtanen, "Sound event detection in real life recordings using coupled matrix factorization of spectral representations and class activity annotations," in *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*. IEEE, 2015, pp. 151–155.

[16] A. Temko and C. Nadeu, "Classification of acoustic events using svm-based clustering schemes," *Pattern Recognition*, vol. 39, no. 4, pp. 682–694, 2006.

[17] J. Salamon and J. P. Bello, "Deep convolutional neural networks and data augmentation for environmental sound classification," *IEEE Signal Processing Letters*, vol. 24, no. 3, pp. 279–283, 2017.

[18] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," *arXiv preprint arXiv:1710.09412*, 2017.

[19] J. F. Gemmeke, D. P. Ellis, D. Freedman, A. Jansen, W. Lawrence, R. C. Moore, M. Plakal, and M. Ritter, "Audio set: An ontology and human-labeled dataset for audio events," in *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*. IEEE, 2017, pp. 776–780.

[20] E. Fonseca, J. Pons Puig, X. Favory, F. Font Corbera, D. Bogdanov, A. Ferraro, S. Oramas, A. Porter, and X. Serra, "Freesound datasets: a platform for the creation of open audio datasets," in *Proceedings of International Society for Music Information Retrieval Conference (ISMIR)*, 2017.

[21] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks." in *CVPR*, vol. 1, no. 2, 2017, p. 3.

[22] E. Fonseca, M. Plakal, F. Font, D. P. Ellis, X. Favory, J. Pons, and X. Serra, "General-purpose tagging of freesound audio with audioset labels: Task description, dataset, and baseline," *arXiv preprint arXiv:1807.09902*, 2018.

[23] K. Choi, D. Joo, and J. Kim, "Kapre: On-gpu audio preprocessing layers for a quick implementation of deep neural network models with keras," *arXiv preprint arXiv:1706.05781*, 2017.

[24] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *ICML*, 2015, p. 448456.

[25] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," *arXiv preprint arXiv:1709.01507*, vol. 7, 2017.

[26] F. Chollet *et al.*, "Keras," https://keras.io, 2015.

[27] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

CONVOLUTIONAL NEURAL NETWORKS AND X-VECTOR EMBEDDING FOR DCASE2018 ACOUSTIC SCENE CLASSIFICATION CHALLENGE

Hossein Zeinali, Lukáš Burget and Jan "Honza" Černocký

Brno University of Technology, Speech@FIT and IT4I Center of Excellence, Czech Republic

ABSTRACT

In this paper, the Brno University of Technology (BUT) team submissions for Task 1 (Acoustic Scene Classification, ASC) of the DCASE-2018 challenge are described. Also, the analysis of different methods on the leaderboard set is provided. The proposed approach is a fusion of two different Convolutional Neural Network (CNN) topologies. The first one is the common two-dimensional CNNs which is mainly used in image classification. The second one is a one-dimensional CNN for extracting fixed-length audio segment embeddings, so called x-vectors, which has also been used in speech processing, especially for speaker recognition. In addition to the different topologies, two types of features were tested: log mel-spectrogram and CQT features. Finally, the outputs of different systems are fused using a simple output averaging in the best performing system. Our submissions ranked third among 24 teams in the ASC sub-task A (task1a).

Index Terms— Audio scene classification, Convolutional neural networks, Deep learning, x-vectors, Regularized LDA

1. INTRODUCTION

This paper deals with the problem of classifying scene or environment (see examples listed in Table 4) based on acoustic clues, which are normally used by humans and animals to understand and react on different environmental condition. Several methods have been proposed for the Acoustic Scene Classification (ASC). Nowadays, most of them are deep learning based. The winner of the last year ASC challenge (i.e. DCASE2017 Task1) used Generative Adversarial Network (GAN) for data augmentation and the combination of Support Vector Machine (SVM) and CNN for classification [1]. The most used network topology in the previous challenges is CNN proven to provide very good performance for ASC [1, 2, 3, 4]. The winner of DCASE2016 Challenge Task1 [5] also used CNN fused with an i-vector based method [6].

This report describes Brno University of Technology (BUT) team submissions for the ASC challenge of DCASE 2018. We proposed two different deep neural network topologies for this task. The first one is a common two-dimensional CNN network for processing audio segments as fixed size two-dimensional images. This network is fed in two ways: with single channel features and 4-channels features. This type of CNN network is useful for detection of audio events invariant to their position in audio signals. The second network topology uses a one-dimensional CNN along the time axis and is used to extract fixed-length embeddings of (possibly variable length) acoustic segments. This architecture has been previously found useful for other speech processing tasks such as speaker recognition [7], where the extracted embeddings were called x-vectors. Therefore, in the rest of the paper, we will also refer to such neural embeddings of acoustic segments as to *x-vectors*. These networks were trained with two feature types: log mel-spectrogram and constant-Q transform (CQT) features. Our submissions are based on fusions of different networks and features trained on the original development data or using additional augmented data.

The current ASC challenge has three sub-tasks: In task1a, participants are allowed to use only the fixed development data for training. Task1b is similar to task1a except that the test files are from different mobile channels. Finally, task1c evaluation data is the same as task1a but additional data is allowed for training. We have participated in task1a only.

2. DATASET

In this work, the DCASE2018 data was used [8]. The dataset consists of recordings from 10 scene classes and was acquired in six large European cities, in different environments in each city. The development set of the dataset consists of 864 segments for each acoustic scene which means a total of 8640 audio segments. The evaluation set was collected in the same cities, but in different environments and has 3600 audio segments. Each segment has an exactly 10-second duration, this is achieved by splitting longer audio recordings from each environment. The dataset includes a predefined validation fold. Each team can also create its own folds, but we used the single official fold for evaluation. The audio segments are 2-channels stereo files, recorded at 48 KHz sampling rate.

3. DATA PROCESSING

3.1. Features

In this work, different features are used in single and multichannel modes. All features are extracted from zero mean audio signals. The main features are log mel-scale spectrogram. For extracting these features, first short time Fourier transform is computed on 40 ms Hamming windowed frames with 20 ms overlap using 2048 point FFT. Next, the power spectrum is transformed to 80 Mel-scale band energies and, finally, log of these energies is taken. The second set of features is obtained as 80-dimensional constant-Q transform of audio signals [9]. This features are extracted using librosa toolbox [10].

We used the features in two modes, single-channel and 4-channels. In single channel mode, the audio signal is first converted to mono and single-channel features are extracted from it (these features are indicated by "M" in the tables). In the 4-channels mode, four sets of features are extracted from the signal similar to [2] (these features are indicated by "LRMS" in the tables). Two feature sets from left (L) and right (R) channels, one from the summation of both channels (i.e. $M = L + R$) and one from the subtraction

of both channels (i.e. $S = L - R$). We use these 4 feature sets as a single input to the CNNs. This mode is similar to multi-channel images (e.g. RGB channels), which are the typical CNN inputs in image classification. In previous works [2, 5], each channel was processed separately and final scores were obtained by fusion of different channel scores. Here, the network tries to use all channels at the same time to use all the available information.

3.2. Data augmentation

Different methods have been proposed for data augmentation in audio processing. Based on the rules of the challenge task1a, external data cannot be used for the data augmentation. Because of this limitation and based on our initial experiments, we decided to use a simple method based on the assumption that a combination of two or more audio segments from the same scene is another sample of that scene with more complex pattern and events. Two new segments were generated for each audio segment as a weighted sum of the audio and several other randomly selected audios from the same scene. This way, we have tripled the amount of training data.

4. CNN TOPOLOGIES

We have used two different CNN topologies for this challenge. The first one is the common two-dimensional CNN known from image processing and the second topology is a one-dimensional CNN for extracting x-vectors – neural network embeddings of audio segment as used, for example, in speaker recognition [7]. Both networks are described in more detail in the following sections.

4.1. Two-Dimensional CNN

We followed the common CNN framework proposed in [4] with some modifications. Table 1 shows the network architecture. The network contains 3 CNN blocks. The first layer is a two-dimensional convolutional layer with 32 filters with kernel size 7×11 and unitary depth and stride in both dimensions. This layer is followed by batch-normalization and *Rectified Linear Unit (ReLU)* activations. The next layer is a max-pooling layer operating over 2×10 non-overlapping rectangles, which is followed by the dropout layer at the end of the CNN block. The output of this block form the input to the next block and so on. The filter and kernel sizes of each layer are shown in Table 1. The last MaxPooling layer in the network operates over the entire time sequence length (i.e. the output of the layer has dimension one for the time axis). The next layer after the third CNN block is a global average pooling (over the frequency axis), which is followed by a batch-normalization layer. Finally, the last layer of the network is a Dense layer (fully connected) with 10 nodes and the softmax activation function. Compared to [4], where only one-channel features were used as the CNN input, we also train another CNN with 4-channel features (as indicated in the first line of Table 1).

4.2. One-dimensional CNN for x-vector extraction

The CNNs extracting x-vectors use one-dimensional convolution along the time. Table 2 shows the network architecture. The network has three parts. The first part operates on the frame-by-frame level and outputs sequence of activation vectors (one for each frame). The second part compresses the frame-by-frame information into a fixed length vector of statistics describing the whole acoustic segment. More precisely, mean and standard deviation of the input activation vectors are calculated over frames. The last part of the network consists of two Dense ReLU layers followed by a Dense softmax layer like in the previous topology. This network has been used in two ways: In the first case, the softmax output is used as before directly for the classification (i.e. we train end-to-end ASC system). In the second case, the x-vectors extracted at the output of the first affine transform after the pooling are used as the input for another classifier.

Table 1: 2-Dimensional CNN topology. BN: Batch Normalization, ReLU: Rectifier Linear Unit. The numbers in the parentheses show the kernel size of convolution layer and the number before BN shows the filter size of the layer. The numbers before MaxPooling show the window size for this layer.

Input $80 \times 500 \times 1$ or $80 \times 500 \times 4$
(7×11) Conv2D(pad=1, stride=1)-32-BN-ReLU (2×10) MaxPooling2D Dropout (0.3)
(7×11) Conv2D(pad=1, stride=1)-64-BN-ReLU (2×5) MaxPooling2D Dropout (0.3)
(7×11) Conv2D(pad=1, stride=1)-128-BN-ReLU (5×10) MaxPooling2D Dropout (0.3)
GlobalAveragePooling2D BatchNormalization
Dense-10-SoftMax

Linear Discriminant Analysis (LDA) transformation is used to precondition the x-vectors for the following ASC classifier (i.e. Cosine similarity classifier). More specifically, it is used to whiten the within-class covariance and possibly reduce the dimensionality of the x-vectors. For this purpose, the conventional LDA can be used, however, the number of preserved dimensions is at most the number of classes minus one (9 in our case). Our previous works in the text-dependent speaker verification [11, 12] and also the ASC experiments here indicate that such dimensionality reduction impacts the performance. For overcoming this limitation, we have proposed to use Regularized version of LDA (RLDA), which enables us to keep as many dimensions as we need. In RLDA, a small fraction of Identity matrix is added to both within and between-class covariance matrices giving the following estimation formulas:

$$\mathbf{S}_w = \alpha\mathbf{I} + \frac{1}{C}\sum_{c=1}^{C}\frac{1}{N_c}\sum_{n=1}^{N_c}(\mathbf{w}_c^n - \overline{\mathbf{w}}_c)(\mathbf{w}_c^n - \overline{\mathbf{w}}_c)^T ,$$

$$\mathbf{S}_b = \beta\mathbf{I} + \frac{1}{C}\sum_{c=1}^{C}(\overline{\mathbf{w}}_c - \overline{\mathbf{w}})(\overline{\mathbf{w}}_c - \overline{\mathbf{w}})^T ,$$

where $\mathbf{I}$ is the identity matrix, C is the total number of classes (i.e. scenes in this case), N_c is the number of training samples in class c, $\mathbf{w}_c^n$ is the n^{th} sample in class c, $\overline{\mathbf{w}}_c = \frac{1}{N_c}\sum_{n=1}^{N_c}\mathbf{w}_c^n$ is the mean of class c, $\overline{\mathbf{w}} = \frac{1}{C}\sum_{n=1}^{C}\overline{\mathbf{w}}_c$ is the mean of the class means and α and β were empirically set to 0.001 and 0.01, respectively. This type of regularization makes the between-class covariance matrix of full rank, which allows us to freely choose the number of dimensions that we wish to preserve after the LDA transformation. In this

Table 2: 1-Dimensional CNN topology for x-vector extraction. BN: Batch Normalization, ReLU:Rectifier Linear Unit.

Input 500×80
(3×1) Conv1D(pad=1, stride=1)-128-ReLU-BN Dropout (0.15)
(3×1) Conv1D(pad=1, stride=1)-128-ReLU-BN Dropout (0.15)
(5×1) Conv1D(pad=1, stride=1)-128-ReLU-BN Dropout (0.15)
(1×1) Conv1D(pad=1, stride=1)-128-ReLU-BN Dropout (0.15)
(1×1) Conv1D(pad=1, stride=1)-256-ReLU-BN
Statistic Pooling, Mean and Standard-Deviation
Dense-128-ReLU-BN (x-vector) Dropout (0.15)
Dense-128-ReLU-BN
Dense-10-SoftMax

work, we reduce the original 128-dimensional x-vectors to 100 dimensions. For more information about RLDA, we refer readers to our previous papers [12, 13].

After applying RLDA, average class x-vectors are estimated on training data and used as class representation vectors. Cosine similarity is calculated between each test x-vector and each class representation vectors and the class with the highest score is selected. Alternatively, these similarity scores are fused with other scores from the CNN outputs for the final decision making.

5. SYSTEMS AND FUSION

In this challenge, we fused outputs of different systems to obtain the final results. For two-dimensional CNNs, both the single-channel and the 4-channels variants are trained on both sets of features, which gives us 4 different classifiers. Further, two CNN for x-vector extraction are trained each on one set of features. These are trained only for the single-channel variant. The softmax outputs of all 6 neural networks are directly used for classification. The two sets of x-vectors produced by the two latter CNNs are further used to construct another two cosine similarity based classifiers.

We trained these systems in two scenarios, the first one using the data without any augmentation and the second one using augmented data. The scores from the resulting 16 systems (8 for each scenario) were fused to form the final submission. We used two different strategies for system fusion: Multiclass logistic regression classifier was trained on the scores from the different systems outputs. FoCal Multiclass toolbox [14] was used for the logistic regression training. As an alternative fusion approach, we simply averaged the scores from the different systems. We used this alternative fusion strategies as we feared that the data available for the logistic regression fusion training might not be sufficient. The logistic regression classifier was trained on the validation set, which was already used for the early-stopping of CNN training and for the model selection (i.e. models performing best on the validation set were selected). Also, this set is rather small, which might lead to over-fitting during the fusion training.

The four final submissions to the challenge were system fusions obtained with the two fusion methods. Each method was used to fuse either 1) all the sub-systems trained only on the augmented data or 2) all the subsystems (i.e. also including the subsystems trained only on the original data).

6. EXPERIMENTAL SETUPS

The experiments reported in this section were mainly carried out on the official challenge validation fold, which divides the development set into two subsets: *training-set* and *evaluation-set*. There are 6122 and 2518 audio segments in each subset, respectively. The training set was further randomly divided into two separate parts with the portions of 70 and 30 percent. The bigger part of the training set was used for network training as well as classifier training (for the cosine distance based method), the smaller part of this set was used for stopping criteria in networks training, model selection and also the fusion training. Finally, the evaluation part was used for reporting the results.

In addition to the results on the development set, some results are reported using Kaggle leaderboard system[1] on a leaderboard set, which has 1200 segments. This set was divided to public and private leaderboard subsets by the organizers and we report the results only for the public subset[2]. In this case, the whole development set was used for training and validation: about 90% randomly selected audio segments of this set were used for training and other segments were used for validation. For the final system training, the same data split was used as for the leaderboard results. The final decisions for 3600 evaluation audio segments was submitted to the challenge website. For final submitted systems, the results on the evaluation set are also reported.

Similar to the baseline system provided by the organizers, our networks training was performed by optimizing the categorical cross-entropy using Adam optimizer [15]. The initial learning rate was set to 0.001 and the network training was early-stopped if the validation loss did not decrease for more than 20 epochs. Then, the training was started again from the best model but now with a reduced learning rate (half value). This training procedure is repeated 3 times until the learning rate reaches 0.00025. The maximum number of epochs and the mini-batch size were set to 200 and 64, respectively.

7. RESULTS

7.1. Comparison of Results

Table 3 reports the public leaderboard results for individual systems as well as several system combinations. We separately report results for the systems using the two different feature sets in order to compare their performance. For the four systems submitted to the challenge, the table also provides the results on the evaluation set.

Comparing the results of the different features, we can see that the mel-spectrogram performs better for ASC task in all cases. However, the fusion of both feature sets improves the performance considerably, which indicates their complementarity.

Generally, feeding the networks with 4-channels features improves the performance as compared to the single-channel variant,

[1] https://www.kaggle.com/c/dcase2018-task1a-leaderboard

[2] The public subset has the same number of audio segments for each class and also is included in the evaluation set. As organizers mentioned, the results on the private subset are not valid because there are different numbers of audio segments per class.

Table 3: Comparison results between different methods and feature types as well as two different fusion strategies and using data-augmentation or not. The star-marks on some fusion systems highlight the systems which were submitted as four final submissions to the challenge. M: single channel feature, LRMS: 4-channels feature, COS: cosine distance, MEL-All: all systems with MEL features and similarly for CQT-all.

Method	Public ACC [%]	Eval. ACC [%]
Baseline system	62.5	61.0
Without data augmentation		
Mel-2D-CNN-M	71.0	
Mel-2D-CNN-LRMS	67.7	
Mel-1D-CNN	65.3	
Mel-x-vector-cos	64.8	
CQT-2D-CNN-M	67.8	
CQT-2D-CNN-LRMS	68.8	
CQT-1D-CNN	60.3	
CQT-x-vector-cos	60.2	
Fusion-Average	**75.0**	
Fusion-FoCal	71.5	
With data augmentation		
Mel-2D-CNN-M	68.2	
Mel-2D-CNN-LRMS	71.3	
Mel-1D-CNN	67.8	
Mel-x-vector-cos	64.7	
CQT-2D-CNN-M	64.8	
CQT-2D-CNN-LRMS	68.5	
CQT-1D-CNN	60.8	
CQT-x-vector-cos	58.2	
Fusion-Average *	**76.8**	**78.1**
Fusion-FoCal *	73.3	75.1
Fusions		
MEL-All-Average	72.5	
CQT-All-Average	71.3	
All-Average *	**77.5**	**78.4**
All-FoCal *	73.0	74.5

especially when more training data is available by the data augmentation. In some cases, this strategy, however, degrades the performance. We believe it should generally improve it, so these cases deserve a further investigation.

When comparing the results from the first and the second sections of Table 3, it is obvious that the augmentation helps in some situations but degrades the performance of other ones. The results are not consistent for all network types. As mentioned before, in four-channel modes the augmentation improves the performance in almost all cases.

The results from the two different fusion strategies show that the simple averaging performs considerably better in all cases. As we expected, the data for the fusion training were not sufficient. The fusion training over-fitted to the validation data and did not generalize well on other datasets.

The results on both leaderboard and evaluation sets show that the fusion of the 8-systems trained on the augmented data already achieves very good performance. When the systems with no data augmentation are also added to the fusion, only slight improvement can be obtained.

Table 4: Comparison results between different scenes of the final fused system.

Scene label	Our system Accuracy [%]	Baseline Accuracy [%]
Airport	91.6	72.9
Bus	71.0	62.9
Metro	78.4	51.2
Metro Station	79.2	55.4
Park	88.4	79.1
Public Square	29.9	40.4
Shopping Mall	77.5	49.6
Street Pedestrian	75.4	50.0
Street Traffic	82.0	80.5
Tram	80.1	55.1
Average	75.3	59.7

7.2. Results on the Official Fold

In this section, the results of the best final system (i.e. All-Average system from Table 3) for each scene are reported. Table 4 shows the performance of the system for each scene separately as well as the overall performance on the official challenge validation fold. The results indicate that our systems perform well for all the scene classes except the *Public Square* class, which deserves a future investigation.

8. CONCLUSIONS

We have described the systems submitted by BUT team to Acoustic Scene Classification (ASC) challenge of DCASE2018. Different systems were designed for this challenge and the final systems were fusions of the output scores from the individual system. A simple score averaging and logistic regression were used for the fusion. The systems included 2-dimensional CNNs with single and 4-channels features, one-dimensional CNNs trained on mel-spectrogram and CQT features. Cosine similarity classifiers were also used to compare x-vectors extracted using the one-dimensional CNNs.

Our future work will include investigations into the failures of the 4-channel CNN variants in some scenarios. We will also experiment with other methods for data augmentation, which, in our opinion, is crucial for the good system performance. Also, we would like to investigate into using bottleneck features for ASC.

9. ACKNOWLEDGMENT

The work was supported by Czech Ministry of Education, Youth and Sports from Project No. CZ.02.2.69/0.0/0.0/16_027/0008371, Czech Ministry of Interior project No. VI20152020025 "DRAPAK", and the National Programme of Sustainability (NPU II) project "IT4Innovations excellence in science - LQ1602". The authors would like to thanks to Mr. Hamid Eghbalzadeh for his valuable discussions. The authors are also tankful to the DCASE organizers for managing the annual challenges, which allows for rapid advances in the ASC technology.

10. REFERENCES

[1] S. Mun, S. Park, D. Han, and H. Ko, "Generative adversarial network based acoustic scene training set augmentation and selection using SVM hyper-plane," DCASE2017 Challenge, Tech. Rep., September 2017.

[2] Y. Han and J. Park, "Convolutional neural networks with binaural representations and background subtraction for acoustic scene classification," DCASE2017 Challenge, Tech. Rep., September 2017.

[3] Z. Weiping, Y. Jiantao, X. Xiaotao, L. Xiangtao, and P. Shaohu, "Acoustic scene classification using deep convolutional neural network and multiple spectrograms fusion," DCASE2017 Challenge, Tech. Rep., September 2017.

[4] R. Hyder, S. Ghaffarzadegan, Z. Feng, and T. Hasan, "BUET bosch consortium (B2C) acoustic scene classification systems for DCASE 2017," DCASE2017 Challenge, Tech. Rep., September 2017.

[5] H. Eghbal-Zadeh, B. Lehner, M. Dorfer, and G. Widmer, "CP-JKU submissions for DCASE-2016: A hybrid approach using binaural i-vectors and deep convolutional neural networks," in *IEEE AASP Challenge on Detection and Classification of Acoustic Scenes and Events (DCASE)*, 2016.

[6] N. Dehak, P. J. Kenny, R. Dehak, P. Dumouchel, and P. Ouellet, "Front-end factor analysis for speaker verification," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 4, pp. 788–798, 2011.

[7] D. Snyder, P. Ghahremani, D. Povey, D. Garcia-Romero, Y. Carmiel, and S. Khudanpur, "Deep neural network-based speaker embeddings for end-to-end speaker verification," in *Spoken Language Technology Workshop (SLT), 2016 IEEE*. IEEE, 2016, pp. 165–170.

[8] A. Mesaros, T. Heittola, and T. Virtanen, "A multi-device dataset for urban acoustic scene classification," in *IEEE AASP Challenge on Detection and Classification of Acoustic Scenes and Events (DCASE)*, 2018.

[9] C. Schörkhuber and A. Klapuri, "Constant-Q transform toolbox for music processing," in *7th Sound and Music Computing Conference, Barcelona, Spain*, 2010, pp. 3–64.

[10] B. McFee, C. Raffel, D. Liang, D. P. Ellis, M. McVicar, E. Battenberg, and O. Nieto, "librosa: Audio and music signal analysis in python," in *Proceedings of the 14th python in science conference*, 2015, pp. 18–25.

[11] H. Zeinali, L. Burget, H. Sameti, O. Glembek, and O. Plchot, "Deep neural networks and hidden Markov models in i-vector-based text-dependent speaker verification," in *Odyssey-The Speaker and Language Recognition Workshop*, 2016, pp. 24–30.

[12] H. Zeinali, H. Sameti, and L. Burget, "HMM-based phrase-independent i-vector extractor for text-dependent speaker verification," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 25, no. 7, pp. 1421–1435, 2017.

[13] H. Zeinali, H. Sameti, and N. Maghsoodi, "SUT submission for NIST 2016 speaker recognition evaluation: Description and analysis," in *Proceedings of the 29th Conference on Computational Linguistics and Speech Processing (ROCLING 2017)*, 2017, pp. 276–286.

[14] N. Brümmer, "FoCal multi-class: Toolkit for evaluation, fusion and calibration of multi-class recognition scorestutorial and user manual," *Software available at http://sites. google. com/site/nikobrummer/focalmulticlass*, 2007.

[15] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

COMBINING HIGH-LEVEL FEATURES OF RAW AUDIO WAVES AND MEL-SPECTROGRAMS FOR AUDIO TAGGING

Marcel Lederle[*]

University of Konstanz
Konstanz, Germany
marcel.lederle@uni.kn

Benjamin Wilhelm[*]

University of Konstanz
Konstanz, Germany
benjamin.wilhelm@uni.kn

ABSTRACT

In this paper, we describe our contribution to Task 2 of the DCASE 2018 Audio Challenge [1]. While it has become ubiquitous to utilize an ensemble of machine learning methods for classification tasks to obtain better predictive performance, the majority of ensemble methods combine predictions rather than learned features. We propose a single-model method that combines learned high-level features computed from log-scaled mel-spectrograms and raw audio data. These features are learned separately by two Convolutional Neural Networks, one for each input type, and then combined by densely connected layers within a single network. This relatively simple approach along with data augmentation ranks among the best two percent in the Freesound General-Purpose Audio Tagging Challenge on Kaggle.

***Index Terms*—** audio-tagging, convolutional neural network, raw audio, mel-spectrogram

1. INTRODUCTION

For humans, it seems to be effortless to associate sounds with events or categories that describe the perceived sound best. However, the complex structure and the large amount of information transmitted through sound makes it particularly difficult to extract that information automatically.

Recognizing a wide variety of sounds has many applications in our today's life. These include surveillance [2, 3, 4], acoustic monitoring [5], and automatic description of multimedia [6]. Due to the diversity of sounds belonging to the same category, a reliable recognition of manifold sound categories is still under ongoing research.

Carefully hand-crafted features such as Mel Frequency Cepstral Coefficients (MFCCs) were the dominant features used for speech recognition [7, 8] and music information retrieval [9], but the trend is now shifting toward deep learning [10]. The approach of manual feature engineering has drawbacks compared to deep learning based methods because it requires considerable effort and expertise to manually create features for a specific purpose. In particular, most of the engineered features, such as MFCCs and spectral centroids [11], are non-task specific, whereas all deep learning approaches are task-specific due to its formulation as a minimization process on task-specific training examples. Since Convolutional Neural Networks (CNNs) have shown a remarkable progress in visual recognition tasks [12] over the last years, it has become common to use CNNs for feature extraction and classification in the audio domain [13, 14]. Several CNN architectures, such as AlexNet [15], VGG [16], ResNet [17], and Inception-v3 [18], have been proposed for image classification, which are also well suited to the task of audio tagging.

The goal of Task 2 of the DCASE 2018 Challenge [1] was to predict the category of an audio clip belonging to one out of 41 heterogeneous classes, such as "Acoustic guitar", "Bark", "Bus" and "Telephone", drawn from the AudioSet Ontology [19]. Training and testing data contain a diverse set of user-generated audio clips from Freesound (https://freesound.org) [1].

In this paper, we mainly focus on building an audio-tagging system that uses both the raw audio data and the corresponding mel-spectrogram rather than ensembling [20] or stacking [21] multiple classifiers.

2. METHOD

Our audio-tagging system comprises two separately trained Convolutional Neural Networks on raw audio and mel-spectrogram, respectively. The learned high-level features are then combined by a densely connected neural network to form the system. In the following, we describe each model in detail.

2.1. CNN on Raw Audio (*cnn-audio*)

For the *cnn-audio* model, we use an architecture similar to common architectures for image classification, like VGG16 [16] or AlexNet [15], but with one-dimensional convolutions and one-dimensional max pooling. As described in Table 1, we use four blocks, each consisting of two convolutional and one max-pooling layer. The number of filters is increased in each consecutive block, while the kernel size is decreased. The pool-size of the max-pooling layers is chosen to quickly reduce the large time dimension. After each block and after the dense layer, we apply batch normalization [22], as experiments have shown that it reduces the training time and increases the model accuracy. To introduce nonlinearities into our network, we apply a ReLU activation function [23] after each convolutional layer and dense layer.

2.2. CNN on Mel-Spectrogram (*cnn-spec*)

The *cnn-spec* model is a two-dimensional convolutional neural network taking mel-spectrograms as input. The architecture is again similar to common image classification architectures and is described in detail in Table 2. As with the one-dimensional model, we apply batch normalization after each block and the dense layer and use the ReLU activation function after convolutional and dense layers.

[*]Both authors contributed equally to this work.

Layer	1sec shape	2sec shape	3sec shape
input	(44100, 1)	(88200, 1)	(132300, 1)
conv1d, 11, 32	(44100, 32)	(88200, 32)	(132300, 32)
conv1d, 11, 32	(44100, 32)	(88200, 32)	(132300, 32)
max-pool1d, 8/16	(5512, 32)	(5512, 32)	(8268, 32)
conv1d, 9, 64	(5512, 64)	(5512, 64)	(8268, 64)
conv1d, 9, 64	(5512, 64)	(5512, 64)	(8268, 64)
max-pool1d, 16	(344, 64)	(344, 64)	(516, 64)
conv1d, 7, 128	(344, 128)	(344, 128)	(516, 128)
conv1d, 7, 128	(344, 128)	(344, 128)	(516, 128)
max-pool1d, 16	(21, 128)	(21, 128)	(32, 128)
conv1d, 5, 256	(21, 256)	(21, 256)	(32, 256)
conv1d, 5, 256	(21, 256)	(21, 256)	(32, 256)
max-pool1d, 16	(1, 256)	(1, 256)	(2, 256)
dense, 512	(512)	(512)	(512)
softmax, 41	(41)	(41)	(41)

Table 1: The architecture of the *cnn-audio* model. Note that for the one-second model, the first max-pooling layer uses a pool-size of 8, while for other models a pool-size of 16 is used.

Layer	1sec shape	2sec shape	3sec shape
input	(128, 170, 1)	(128, 300, 1)	(128, 400, 1)
conv2d, 4×4, 64	(128, 170, 64)	(128, 300, 64)	(128, 400, 64)
conv2d, 4×4, 64	(128, 170, 64)	(128, 300, 64)	(128, 400, 64)
max-pool2d, 1×1/2×2	(128, 170, 64)	(64, 150, 64)	(64, 200, 64)
conv2d, 4×4, 64	(128, 170, 64)	(64, 150, 64)	(64, 200, 64)
max-pool2d, 2×2	(64, 85, 64)	(32, 75, 64)	(32, 100, 64)
conv2d, 3×3, 128	(64, 85, 128)	(32, 75, 128)	(32, 100, 128)
max-pool2d, 2×4	(32, 21, 128)	(16, 18, 128)	(16, 25, 128)
conv2d, 3×3, 128	(32, 21, 128)	(16, 18, 128)	(16, 25, 128)
max-pool2d, 2×2	(16, 10, 128)	(8, 9, 128)	(8, 12, 128)
conv2d, 3×3, 256	(16, 10, 256)	(8, 9, 256)	(8, 12, 256)
max-pool2d, 2×2	(8, 5, 256)	(4, 4, 256)	(4, 6, 256)
conv2d, 3×3, 256	(8, 5, 256)	(4, 4, 256)	(4, 6, 256)
max-pool2d, 2×2	(4, 2, 256)	(2, 2, 256)	(2, 3, 256)
dense, 256	(256)	(256)	(256)
softmax, 41	(41)	(41)	(41)

Table 2: The architecture of the *cnn-spec* model. Note that the first max-pooling layer does not exist in the case of the one-second model.

The mel-spectrogram is extracted using librosa [24] with the original sampling frequency of 44.1 kHz, 2048 FFT points, 128 mel-bins, and a hop-length of 256. The amplitude of the mel-spectrogram is scaled logarithmically, and the scaled mel-spectrogram is resized in time dimension to fit the model input size.

2.3. Joining CNNs (*cnn-comb*)

We remove the softmax and dense layer of both the trained *cnn-audio* and *cnn-spec* model and then concatenate the output features of the previous layer of both models so as to join them. The concatenated features are then connected to a densely connected neural network with four hidden layers. The hidden dense layers have 512, 256, 256, and 128 neurons, respectively. The complete model is illustrated in Figure 1.

We train *cnn-audio* and *cnn-spec* from scratch. Afterward, the weights of these models are transferred to the *cnn-comb* model and only the newly added dense layers are trained. The splitting of the training of *cnn-comb* into three steps, facilitates the procedure.

2.4. Data Augmentation

To prevent our model from overfitting, we make use of extensive data augmentation during training (time shifting, cropping, padding, and blending clips of same and different categories). Each of these augmentation techniques is applied to the raw audio wave and the mel-spectrogram. In the remaining section, we explain the augmentation methods based on the raw audio wave.

First, we apply a uniformly random time shift to the audio clip. To ensure that the audio clips fit the size of the model input, crops are taken from too long audio files and too short audio files are padded. For audio clips that are longer than the model input size, we use a crop with the size of the model input taken from a random position. If an audio sample fits multiple times (n times) in the input size, it is replicated such that it appears $k \in \{1, \ldots, n\}$ times with a probability of $1/n$, and the remaining space before, after, and in between replications is filled with zeros.

Additionally, we enhance the robustness of our model by blending multiple audio clips of same or different categories. This method is referred to as mixup [25]. We blend them by assigning a random weight to each sample (weights sum up to one) and taking the weighted sum. If the blended samples are of the same class, the model should still predict the common class for the newly generated training sample. In the other case, if the blended samples are of distinct classes, the model is trained to predict the weight of each included class.

While the spectrogram is computed in advance, the computationally inexpensive data augmentation techniques can be computed on-the-fly during training. This saves disk-space and guarantees a large amount of diverse training data.

2.5. Implementation Details

We implemented the described method using Keras [26] in Python.

To monitor overfitting and the model performance during training, we exclude a part of the training data as validation data. To still make use of all training data, we train five models on stratified folds of the training data such that each training example is used once for validation and four times for training. For the final prediction, we accumulate the predictions of all five models using the geometric mean.

All models are trained using the Adam optimizer [27] with a fixed learning rate of 0.001 and a mini-batch size of 32 for a maximum of 300 epochs, but stopping earlier if the validation loss hasn't improved for 35 epochs. We use the categorical cross-entropy loss function and weight the loss according to the distribution of training examples per class, thereby ensuring that the models pay more attention to samples from an under-represented class.

Because many clips contain silence, we cut off silent parts at

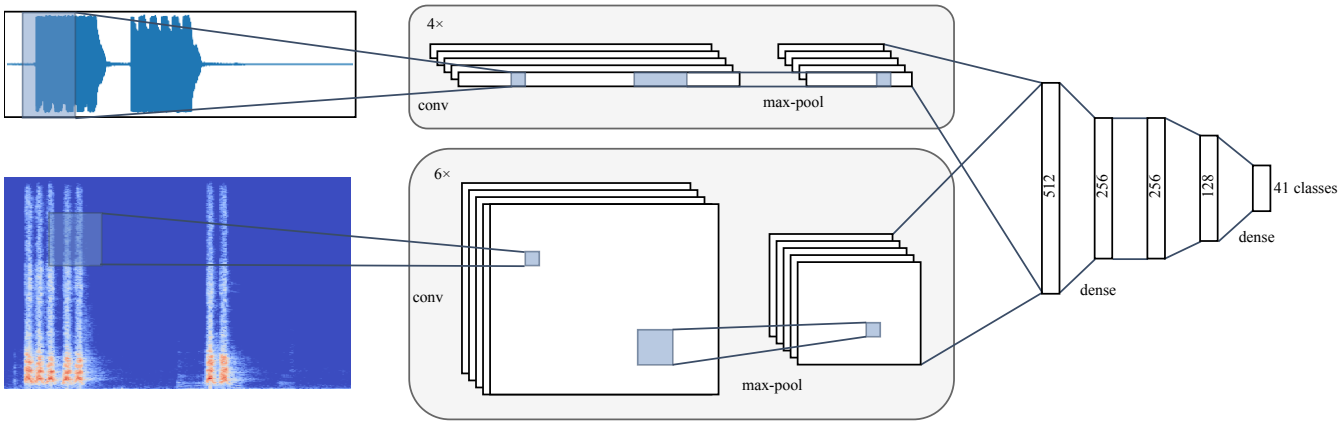


Figure 1: Illustrated architecture of the complete model.

the beginning and at the end of an audio clip that do not exceed a volume of 40 decibel.

When predicting the test data, we have to take the varying length of audio files into account. It is not sufficient to only predict on one crop of too long tracks because important features might not be present in the selected crop. Therefore, we run the inference on many crops of the audio file with a step size of 5120 frames, which is approximately 0.12 seconds. For too short audio tracks, the model might be able to better recognize class-specific features in certain parts of the input. Therefore, we generate multiple inputs by padding the audio file with zeros such that the real audio appears at different positions in the input. Again, we use a step size of 5120 frames. Multiple predictions for one audio file are combined by means of the geometric mean.

3. EVALUATION

3.1. Dataset

We evaluate our method on the dataset provided for Task 2 of the DCASE 2018 Challenge [1], which comprises 9473 training and 1600 test samples. The test data set has been manually verified, whereas the training data features labels of different reliability. Each mono audio file has a bit-depth of 16, a sampling rate of 44.1 kHz, and is associated with one out of 41 classes of the AudioSet Ontology [19].

The class distribution of both training and test set is not balanced and ranges from 94 to 300 and from 25 to 110 samples per class, respectively. The duration of the shortest audio file is 300 ms and 30 seconds for the longest clip, while the average length is 6.8 seconds for the train set and 5.2 seconds for the test set.

3.2. Metric

The Challenge uses mean Average Precision at three (mAP@3) for evaluating test results, which allows up to three predictions per audio clip. Full credit is given if the first prediction matches the label of the clip, while less credit is given if one of the other predictions is correct. The evaluation metric is defined as

$$\text{mAP@3} = \frac{1}{U} \sum_{i=1}^{U} \sum_{j=1}^{\min(3,n_i)} \frac{[\![y_{ij} = \hat{y}_i]\!]}{j},$$

Model	Crop length	Public score (mAP@3)	Private score (mAP@3)	Total (mAP@3)
cnn-audio	1sec	0.920	0.888	0.894
	2sec	0.921	0.884	0.891
	3sec	0.935	0.889	0.898
cnn-spec	1sec	0.930	0.923	0.924
	2sec	0.950	0.928	0.932
	3sec	0.935	0.930	0.931
cnn-comb	1sec	0.955	0.939	0.942
	2sec	**0.966**	**0.944**	**0.948**
	3sec	0.956	**0.944**	0.946

Table 3: Evaluation results of the individual models on the public (301 samples), private (1299 samples), and full test set.

where U is the total number of scored audio files, y_{ij} is the predicted label for file i at position j, $\hat{y}_i$ is the ground-truth label for file i, n_i is the total number of predicted labels for file i, $[\![\texttt{True}]\!] = 1$, $[\![\texttt{False}]\!] = 0$. No label may be predicted multiple times for one audio file.

3.3. Results

We have trained all models on inputs of one, two, and three seconds, as described in Section 2.5, and evaluated the model performance on the test set (see Table 3). We have observed that for each crop length, the combined model performs significantly better compared to models with a single input. Combined models with an input size of two and three seconds, perform best and rank in the upper two percent on the Private Leaderboard on Kaggle.

Additionally, we determined the per-category mAP@3 on the complete test set showing that some classes are more challenging to predict than others (see Table 4). Our model primarily struggles with the classes “Squeak”, “Telephone” and “Fireworks”, but it still beats the baseline system [1] in every per-category mAP@3 score.

To verify that the performance gain of the combined model results from combining the extracted high-level features from both models, we compare *cnn-audio* and *cnn-spec* to the *cnn-comb*

Name	samples	time	mAP@3	Name	samples	time	mAP@3	Name	samples	time	mAP@3
Acoustic_guitar	300	52.2	0.893	Electric_piano	150	25.5	1.000	Microwave_oven	146	25.1	0.966
Applause	300	58.2	1.000	Fart	300	18.6	0.944	Oboe	299	15.3	0.976
Bark	239	44.6	0.982	Finger_snapping	117	5.9	1.000	Saxophone	300	33.7	0.942
Bass_drum	300	12.8	1.000	Fireworks	300	48.2	0.786	Scissors	95	15.7	0.927
Burping_or_eructation	210	11.7	1.000	Flute	300	46.2	1.000	Shatter	300	26.1	0.960
Bus	109	28.4	0.953	Glockenspiel	94	8.4	0.856	Snare_drum	300	17.9	0.912
Cello	300	37.3	0.951	Gong	292	41.8	0.968	Squeak	300	38.2	0.603
Chime	115	23.8	0.891	Gunshot_or_gunfire	147	11.1	0.950	Tambourine	221	10.1	0.975
Clarinet	300	34.7	0.991	Harmonica	165	18.6	0.970	Tearing	300	38.7	0.981
Computer_keyboard	119	23.0	1.000	Hi-hat	300	18.6	0.957	Telephone	120	16.2	0.788
Cough	243	22.4	1.000	Keys_jangling	139	18.8	0.929	Trumpet	300	28.3	0.959
Cowbell	191	10.9	1.000	Knock	279	19.6	0.957	Violin_or_fiddle	300	26.6	0.986
Double_bass	300	16.9	0.946	Laughter	300	36.3	0.974	Writing	270	48.3	0.948
Drawer_open_or_close	158	18.0	0.925	Meow	155	18.7	1.000				

Table 4: Per category mAP@3 score of the *cnn-comb* 2sec model on the full test set and the number of samples along with time in minutes of the respective class in the train set.

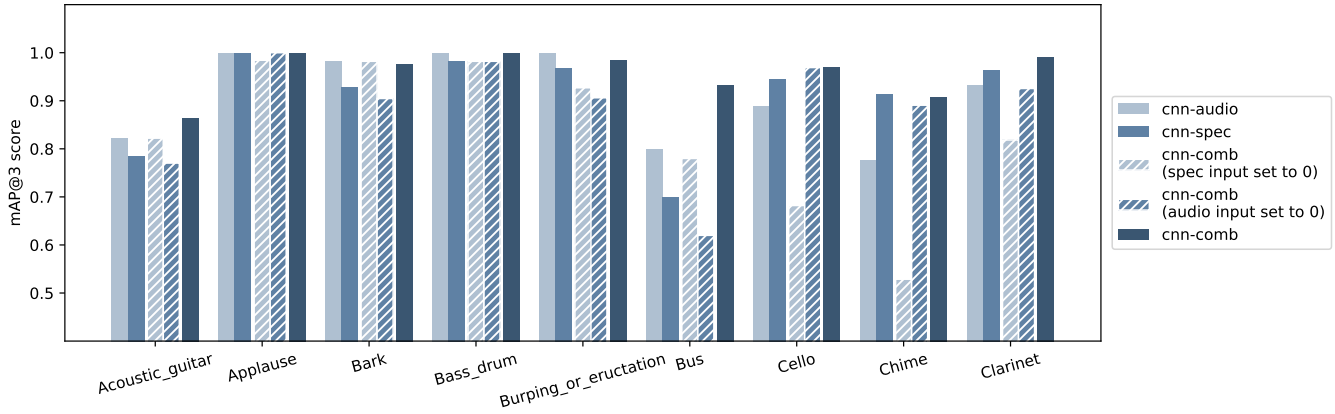


Figure 2: Comparison of per-category scores of single input models, combined models with one input alternately set to zero, and the combined model with both inputs. The mAP@3 score is reported on a single fold for each model.

model, where one of its inputs is set to zero (See Figure 2).

For categories in which the single *cnn-audio* model outperforms the single *cnn-spec* model, the combined model performs better if the audio input is present and not set to zero. Otherwise, if *cnn-spec* outperforms *cnn-audio*, *cnn-comb* has a higher score if the mel-spectrogram is present and not set to zero. Setting either the mel-spectrogram or the audio wave to zero forces the *cnn-comb* model to make predictions based on a single input.

cnn-comb performs equally or better than *cnn-comb* with one single input set to zero because it has learned to utilize meaningful high-level features of both inputs jointly, which are not given by zeroed inputs. For the same reason, *ccn-comb* with one zeroed input usually performs worse than the corresponding model with a single input.

We conclude that *cnn-comb* makes use of the learned high-level features from both *cnn-audio* and *cnn-spec*, but it focuses more on features belonging to the superior single model.

4. CONCLUSION

In this paper, we have proposed a method for audio-tagging that extends current Convolutional Neural Network approaches that only make use of a frequency representation by adding a second input that incorporates the raw audio wave. Adding the additional input, has improved the mAP@3 score significantly. We have demonstrated the capabilities of our model by competing in the Freesound General-Purpose Audio Tagging Challenge on Kaggle and ranking in the top two percent of all participants.

5. ACKNOWLEDGMENT

We thank Christian Borgelt and Christoph Doell for motivating us to take part in the Kaggle competition and Christoph Doell for his valuable comments on the manuscript.

6. REFERENCES

[1] E. Fonseca, M. Plakal, F. Font, D. P. Ellis, X. Favory, J. Pons, and X. Serra, "General-purpose tagging of freesound audio with audioset labels: Task description, dataset, and baseline," *arXiv preprint arXiv:1807.09902*, 2018.

[2] A. Harma, M. F. McKinney, and J. Skowronek, "Automatic surveillance of the acoustic activity in our living environment," in *Multimedia and Expo, 2005. ICME 2005. IEEE International Conference on*. IEEE, 2005, pp. 4–pp.

[3] M. Crocco, M. Cristani, A. Trucco, and V. Murino, "Audio surveillance: a systematic review," *ACM Computing Surveys (CSUR)*, vol. 48, no. 4, p. 52, 2016.

[4] G. Valenzise, L. Gerosa, M. Tagliasacchi, F. Antonacci, and A. Sarti, "Scream and gunshot detection and localization for audio-surveillance systems," in *Advanced Video and Signal Based Surveillance, 2007. AVSS 2007. IEEE Conference on*. IEEE, 2007, pp. 21–26.

[5] S. Goetze, J. Schroder, S. Gerlach, D. Hollosi, J.-E. Appell, and F. Wallhoff, "Acoustic monitoring and localization for social care," *Journal of Computing Science and Engineering*, vol. 6, no. 1, pp. 40–50, 2012.

[6] Q. Jin and J. Liang, "Video description generation using audio and visual cues," in *Proceedings of the 2016 ACM on International Conference on Multimedia Retrieval*. ACM, 2016, pp. 239–242.

[7] C. Ittichaichareon, S. Suksri, and T. Yingthawornsuk, "Speech recognition using MFCC," in *International Conference on Computer Graphics, Simulation and Modeling (ICGSM'2012) July*, 2012, pp. 28–29.

[8] B. Logan *et al.*, "Mel frequency cepstral coefficients for music modeling." in *ISMIR*, vol. 270, 2000, pp. 1–11.

[9] G. Tzanetakis and P. Cook, "Musical genre classification of audio signals," *IEEE Transactions on speech and audio processing*, vol. 10, no. 5, pp. 293–302, 2002.

[10] O. Abdel-Hamid, L. Deng, and D. Yu, "Exploring convolutional neural network structures and optimization techniques for speech recognition." in *Interspeech*, vol. 2013, 2013, pp. 1173–5.

[11] A. J. Eronen, V. T. Peltonen, J. T. Tuomi, A. P. Klapuri, S. Fagerlund, T. Sorsa, G. Lorho, and J. Huopaniemi, "Audio-based context recognition," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 14, no. 1, pp. 321–329, 2006.

[12] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, *et al.*, "ImageNet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

[13] O. Abdel-Hamid, A.-r. Mohamed, H. Jiang, L. Deng, G. Penn, and D. Yu, "Convolutional neural networks for speech recognition," *IEEE/ACM Transactions on audio, speech, and language processing*, vol. 22, no. 10, pp. 1533–1545, 2014.

[14] K. J. Piczak, "Environmental sound classification with convolutional neural networks," in *Machine Learning for Signal Processing (MLSP), 2015 IEEE 25th International Workshop on*. IEEE, 2015, pp. 1–6.

[15] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[16] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[17] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[18] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2818–2826.

[19] J. F. Gemmeke, D. P. Ellis, D. Freedman, A. Jansen, W. Lawrence, R. C. Moore, M. Plakal, and M. Ritter, "Audio Set: An ontology and human-labeled dataset for audio events," in *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*. IEEE, 2017, pp. 776–780.

[20] L. Breiman, "Bagging predictors," *Machine learning*, vol. 24, no. 2, pp. 123–140, 1996.

[21] S. Džeroski and B. Ženko, "Is combining classifiers with stacking better than selecting the best one?" *Machine learning*, vol. 54, no. 3, pp. 255–273, 2004.

[22] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[23] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th international conference on machine learning (ICML-10)*, 2010, pp. 807–814.

[24] B. McFee, M. McVicar, S. Balke, C. Thomé, C. Raffel, O. Nieto, E. Battenberg, D. Ellis, R. Yamamoto, J. Moore, R. Bittner, K. Choi, F.-R. Stöter, S. Kumar, S. Waloschek, Seth, R. Naktinis, D. Repetto, C. F. Hawthorne, C. Carr, hojinlee, W. Pimenta, P. Viktorin, P. Brossier, J. F. Santos, JackieWu, Erik, and A. Holovaty, "librosa/librosa: 0.6.0," Feb. 2018. [Online]. Available: https://doi.org/10.5281/zenodo.1174893

[25] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," in *International Conference on Learning Representations*, 2018.

[26] F. Chollet *et al.*, "Keras," https://keras.io, 2015.

[27] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

GENERAL-PURPOSE AUDIO TAGGING FROM NOISY LABELS USING CONVOLUTIONAL NEURAL NETWORKS

Turab Iqbal, Qiuqiang Kong, Mark D. Plumbley, Wenwu Wang

Centre for Vision, Speech and Signal Processing, University of Surrey
{t.iqbal, q.kong, m.plumbley, w.wang}@surrey.ac.uk

ABSTRACT

General-purpose audio tagging refers to classifying sounds that are of a diverse nature, and is relevant in many applications where domain-specific information cannot be exploited. The DCASE 2018 challenge introduces Task 2 for this very problem. In this task, there are a large number of classes and the audio clips vary in duration. Moreover, a subset of the labels are noisy. In this paper, we propose a system to address these challenges. The basis of our system is an ensemble of convolutional neural networks trained on log-scaled mel spectrograms. We use preprocessing and data augmentation methods to improve the performance further. To reduce the effects of label noise, two techniques are proposed: loss function weighting and pseudo-labeling. Experiments on the private test set of this task show that our system achieves state-of-the-art performance with a mean average precision score of 0.951.

Index Terms— Audio classification, convolutional network, recurrent network, deep learning, data augmentation, label noise

1. INTRODUCTION

Audio tagging is a classification problem that is concerned with categorizing audio clips based on the presence of sound events. These events could be domestic sounds such as a telephone ringing, outdoor sounds such as a car passing by, and anything else that may be relevant to an application. Historically, classifiers have relied on domain-specific techniques to achieve good performance, such as in speech recognition [1] and music information retrieval [2]. However, with new applications such as smart homes [3] and smart cities [4], there is growing interest in general-purpose audio classifiers.

The Detection and Classification of Acoustic Scenes and Events (DCASE) [5] is a recurring challenge with several tasks pertaining to audio classification. DCASE 2018 introduces Task 2 [6], which poses the problem of general-purpose audio tagging. This task uses a subset of the FSD dataset [7], and is comprised of 41 audio classes with labels from Google's AudioSet ontology [8]. As this is a high number of classes, it demands good discriminative abilities from the classifier. A training set of 9473 labeled examples is provided in order to use supervised learning methods. However, approximately $60\,\%$ of the labels are unverified. Of these unverified labels, at least $65\,\%$ to $70\,\%$ of the labels are correct. The presence of incorrectly-labeled examples can negatively affect training, so it is important that the learning algorithm is robust with respect to such examples. Another property of this dataset is that the duration of the audio clips varies from $0.3\,\mathrm{s}$ to $30\,\mathrm{s}$. This is a problem because many training models expect a fixed-length input.

In this paper, we propose a system to address these challenges. The system we develop is based on a number of convolutional neural networks trained on log-mel spectrograms. We investigate different architectures of convolutional neural networks and show that they complement each other by ensembling predictions using a technique called stacking. We also use preprocessing and data augmentation methods to improve the performance further. To reduce the effects of incorrect labels, two ideas are proposed: loss function weighting and pseudo-labeling.

This paper is organized as follows. Section 2 introduces related work. Section 3 details the preprocessing and feature extraction methods. Section 4 proposes the convolutional neural networks and training methodology. Section 5 presents the experiments and results. Lastly, Section 6 concludes with a summary.

2. RELATED WORKS

Deep neural networks have recently become a popular choice for audio classification due to their leading performance in many tasks, including general-purpose audio tagging [9]. Most, if not all, of the neural network architectures that achieve state-of-the-art results are based on convolutional neural networks (CNN) [10, 11, 12]. Hybrid architectures such as convolutional recurrent neural networks (CRNN) have also been adopted with great success [13, 14, 15], but more for tasks that also require localization. Given a training model, an effective way to improve the performance further is to use data augmentation, as observed in [16] and [17]. We use both CNNs and CRNNs with data augmentation, but also consider label noise.

Learning from noisy labels is a problem that has several decades of research behind it [18]. It was shown in [19] that neural networks can converge to near-zero loss on training sets even when the labels are completely random. This is problematic because it suggests that incorrect labels can have a significant impact on the generalization performance of a neural network. One way to mitigate the effects of label noise is to apply a correction to the loss function as in [20], which requires estimating a noise transition matrix. In [21], this is done by incorporating an additional layer in the neural network. We also modify the loss function, but make simpler assumptions and hence propose a simpler method. Furthermore, we combine this with a technique called pseudo-labeling.

3. PREPROCESSING AND FEATURE EXTRACTION

3.1. Silence Removal

In the dataset of Task 2, we observed that there is little background noise present in the audio clips. However, some of the clips contain segments of silence. Long sequences of silence are not characteristic of the sounds themselves, and only indicate the start or end of sounds. For this reason, we include in our pipeline an algorithm to extract the non-silent segments of the audio signal. This discards the silence and means that separate segments can be considered as separate inputs.

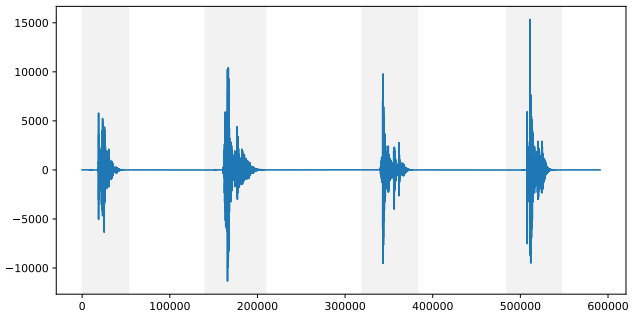


Figure 1: Illustration of the silence removal process on the file "071e836c.wav", which is 13 s in length. The non-silent segments that are extracted are highlighted in gray. These segments would be considered as separate inputs.

Table 1: Log-mel spectrogram parameters

Parameter	Configuration A	Configuration B
Sample rate	32 000 Hz	32 000 Hz
Window size	1024	512
Hop size	512	256
Mel bands	64	64

To detect silence, we segment the audio signal into frames and threshold each frame's root mean square (RMS) energy. We define "silence" to include very quiet background noise, so it does not have to match the threshold of hearing. In our algorithm, a non-silent segment is the span of non-silent frames that are within some proximity (they do not have to be contiguous), and also includes some excess silence to prevent over-cropping. This is exemplified in Figure 1, where four segments are highlighted.

3.2. Feature Extraction

Following silence removal, the extracted segments are considered as separate inputs. These are downsampled to 32 kHz and transformed into log-scaled mel-frequency (log-mel) spectrograms. Log-mel features have been shown to outperform traditional representations such as mel-frequency cepstrum coefficients (MFCCs) in modern neural network architectures [22], which benefit from the additional information and do not need de-correlated inputs [23].

We use two sets of parameters to extract the log-mel features – a standard configuration and a narrow-band configuration – the values of which are given in Table 1. We selected these values based on experiments on a validation set. In our experiments, these configurations produced complementing results. Indeed, the different resolutions appear to capture different characteristics.

After feature extraction, each feature vector is partitioned into chunks of a fixed size, which resolves the problem of varying clip durations. When the length of the feature vector is less than the chunk length, it is padded. When it is greater, but not evenly divisible by the chunk length, an additional chunk is added to align with the end of the feature vector so that it includes the remainder.

The chunk size is an important parameter, as a chunk that is too short may not encompass a sound in its entirety. On the other hand, a chunk that is too long will mostly contain padding data for short audio segments. We choose a chunk size of 128×64, where the first axis is the temporal dimension. This corresponds to 2 s chunks and 1 s chunks for configurations A and B, respectively (cf. Table 1). Approximately 80 % of the segments in the training set are more than 1 s in duration, and 60 % are more than 2 s.

4. TRAINING AND INFERENCE

In order to utilize the training set that is provided for this task, we use two types of neural networks: CNNs and CRNNs. For each type, we also use two variants: one using standard convolutions and another using gated convolutions. Since there are two log-mel configurations, this gives eight training models in total. In the subsections to follow, we describe the architectures, learning with label noise, data augmentation, and ensembling.

4.1. Neural Network Architectures

The neural network architectures are outlined in Table 2. Beginning with the standard CNN, it is essentially equivalent to the "VGG13" network proposed in [24], hence the name. Each convolutional block consists of two convolutional layers followed by a max pooling layer that halves each spatial dimension. The convolutional layers use a ReLU activation function [25] as well as batch normalization [26] as a form of regularization. After the convolutional blocks, global average pooling is applied, i.e. each feature map is averaged across both dimensions. Finally, a densely-connected softmax layer is used to generate the predictions.

The CRNN architecture is an extension of VGG13. Instead of applying global average pooling after the convolutions, only the frequency dimension is averaged so that temporal information is preserved. A bidirectional recurrent layer [27] is then applied to output a vector, $\mathbf{s}_t$, for each time step $t \in [1, T]$. Averaging these vectors gives $\mathbf{s} = \frac{1}{T}\sum_{t=1}^{T} \mathbf{s}_t$, which is the output prior to the softmax layer. The motivation for using a recurrent layer is to learn the temporal dynamics of the input [13]. In our experiments, this improved the overall performance by up to 0.5 %.

The other two architectures are GCNN and GCRNN, which are variants of VGG13 and CRNN, respectively. The difference is that each convolutional layer is replaced with a gated convolutional layer [28]. The idea of a gated layer is inspired by the gating mechanisms found in recurrent neural networks [29, 30], and is used to control the information that is propagated to deeper layers. This mechanism has been shown to produce good results for similar tasks [15].

4.2. Learning from Noisy Labels

The presence of unverified labels poses a problem for learning, as neural networks are susceptible to overfitting on incorrectly-labeled examples [19]. On the other hand, training with verified examples *only* means that a large number of unverified labels that are otherwise correct are discarded. Our tests showed that performance dropped by up to 5 % when a model was trained on verified examples only. Therefore, we propose two techniques.

The first technique is to weight the training loss function such that its magnitude is lowered for unverified examples. The rationale is that if an example is incorrectly labeled, the computed loss will be incorrect and should be disregarded. Of course, we do not know whether it is correct or not if the label is unverified. Let $\eta \in (0, 1)$ be the weight applied to unverified examples. Given a loss function, $L(\mathbf{y}, \hat{\mathbf{y}})$, the weighted loss function is then given by

$$\tilde{L}(\mathbf{y}, \hat{\mathbf{y}}) \coloneqq (\eta \cdot \mathbb{1}_{\mathcal{X}_N}(\mathbf{x}) + \mathbb{1}_{\mathcal{X}_N^c}(\mathbf{x}))L(\mathbf{y}, \hat{\mathbf{y}}), \qquad (1)$$

Table 2: Description of the neural networks. Each convolutional layer uses a receptive field size of 3, as in [24]. "GLU" refers to the use of gated linear units, as described in [28]. "Bi-GRU" refers to using bidirectional gated recurrent units [30].

Feature Size	CRNN	GCRNN
128×64	Log-mel spectrogram	
	$2 \times \{\text{conv } 64, \text{ReLU}\}$	$2 \times \{\text{conv } 64, \text{GLU}\}$
64×32	2×2 Max Pooling	
	$2 \times \{\text{conv } 128, \text{ReLU}\}$	$2 \times \{\text{conv } 128, \text{GLU}\}$
32×16	2×2 Max Pooling	
	$2 \times \{\text{conv } 256, \text{ReLU}\}$	$2 \times \{\text{conv } 256, \text{GLU}\}$
16×8	2×2 Max Pooling	
	$2 \times \{\text{conv } 512, \text{ReLU}\}$	$2 \times \{\text{conv } 512, \text{GLU}\}$
8×4	2×2 Max Pooling	
	$2 \times \{\text{conv } 512, \text{ReLU}\}$	$2 \times \{\text{conv } 512, \text{GLU}\}$
4×2	2×2 Max Pooling	
	Bi-GRU, 512, ReLU (optional)	
	Global Average Pooling	
	Softmax (41 Classes)	

where $\mathcal{X}_N$ is the subset of the training set that is unverified and $\mathcal{X}_N^c$ is the complement. The parameter η can be considered as the confidence that the unverified labels are correct. It can be be determined using a validation set or set to 1 minus the noise rate.

The second technique, known as pseudo-labeling, is to relabel the unverified examples prior to training using a previously-trained classifier. For pseudo-labeling to be effective, the error rate of the classifier should be lower than the noise rate of the original labels, because the error rate can be considered as the new noise rate. The previously-trained classifier in this case is just the same system described in this paper but without pseudo-labeling.

Another form of pseudo-labeling that we propose is to "promote" examples from unverified to verified by corroborating the label with the predictions of the previously-trained classifier. The examples if promoted if the classifier prediction agrees with confidence greater than τ. The confidence threshold, τ, should be high enough that the false positive rate of the classifier is low.

4.3. Data Augmentation

When training a neural network, overfitting is a common problem in which the network learns to predict the training examples with very high accuracy but cannot generalize to new data. This is likely to occur with smaller datasets, and was found to be the case with the dataset of Task 2 by verifying on a validation set. To prevent this, data augmentation is a popular approach [31] that increases the size of the training set without manual intervention. In our work, we use a method called *mixup* [32] to create new examples.

Mixup is a method that generates new data during training by randomly mixing pairs of inputs and their associated target values. Consider a pair of inputs, $\mathbf{x_1}$ and $\mathbf{x_2}$, and their one-hot-encoded target values, $\mathbf{y_1}$ and $\mathbf{y_2}$. To mix these, a parameter, $\lambda \in (0, 1)$, is used to create convex combinations.

$$\mathbf{x} = \lambda \mathbf{x}_1 + (1 - \lambda)\mathbf{x}_2. \tag{2}$$

$$\mathbf{y} = \lambda \mathbf{y}_1 + (1 - \lambda)\mathbf{y}_2. \tag{3}$$

The output, $(\mathbf{x}, \mathbf{y})$, is then used as the training example rather than the original examples. If loss function weighting is used, $w(\mathbf{x}) \coloneqq \eta \cdot \mathbb{1}_{\mathcal{X}_N}(\mathbf{x}) + \mathbb{1}_{i\mathcal{X}_N^c}(\mathbf{x})$ should also be mixed.

Table 3: Training parameters

Parameter	Value
Batch size	128
Learning rate (LR)	0.0005
LR decay factor	0.9
LR decay rate	2

In the original paper [32], a different value of λ is used for each mini-batch by sampling from a beta distribution, $\mathrm{B}(\alpha, \alpha)$. The hyperparameter α controls the distribution's shape, where a value of 1.0 reduces it to a uniform distribution. A lower value of α will be more likely to produce values of λ that are closer to 0 and 1, which weakens the effect of mixup.

Mixup has been used before for audio classification and has been shown to be beneficial [17]. We ultimately followed the method used in [32], but also experimented with a variation. This involves constraining some or all of the example pairs to belong to the same class. Unfortunately, this did not improve the performance of our system and actually worsened it in the extreme case of constraining all the pairs. This suggests that mixing inter-class examples is an important aspect of mixup's success.

5. EXPERIMENTS

5.1. System Setup

To train the various models, the training set was split into five cross-validation folds, ensuring that there was a similar number of verified examples in each fold. The cross-entropy function was used as the training loss and Adam [33] was used as the gradient descent algorithm. Refer to Table 3 for the values of the hyperparameters. Decay rate is the number of epochs until the learning rate is decayed.

For loss function weighting, we used a weight of $\eta = 0.7$ when pseudo-labeling was not used (to determine the pseudo-labels in the first place) and $\eta = 0.9$ when it was. When promoting labels, we used a confidence threshold of $\tau = 0.7$. For mixup, the parameter α was set to 1.0. All of the hyperparameters were selected based on evaluations on a validation set containing verified labels only. This includes the parameters in Table 3 too.

In terms of generating the predictions, the top four epochs were selected based on performance on the validation set. The metric used was the mean average precision (MAP) score. Recalling that the inputs to the neural networks are chunks, and that the chunks are from sections of the original audio clip (cf. Section 3), the chunk predictions need to be merged to produce clip-level predictions. This was achieved using the geometric mean, as this is less sensitive to outliers than the arithmetic mean. With the clip-level predictions, the top four epochs were merged using the arithmetic mean.

5.2. Ensembling

To combine the predictions of the different models, we used an ensembling method known as stacking [34, 35, 36]. In this method, the base model predictions are used as features to train a second-level classifier. The output of a base model is an $N \times K$ vector of probabilities, where N is the number of data samples and $K = 41$ is the number of classes. By concatenating the outputs of the models, the result is an $N \times 8K$ vector; this is the input of the new classifier.

Table 4: Training set results. Only the results of configuration A models are given to highlight the difference in architectures.

Model	MAP@3
VGG13	0.950
GCNN	0.951
CRNN	0.952
GCRNN	0.958
Arithmetic Mean	0.968
Stacking	0.972

Table 5: Test set results comparing the 8-model stacked ensemble with a 4-model version that excludes VGG13 and CRNN models. The results of the competition's baseline system is also included.

Model	Private	Public	All
Baseline	0.694	0.704	–
4-Model Stacking	0.948	0.956	0.950
8-Model Stacking	0.951	0.961	0.953

As the validation sets constitute the training set, the validation set predictions were used to generate the features for the training set. Similarly, the test set predictions were used as the test set features. We used logistic regression with an L_2 penalty as the second-level classifier. It was configured to use class weights to compensate for class imbalance and sample weights as described in Section 4.2.

5.3. Results

To assess the performance of our system, we evaluated the training set and test set predictions. It was possible to evaluate the training set predictions because we used cross-validation folds. We only evaluated the manually-verified training examples. The test set was split into a public set and a private set by the challenge organizers. Therefore, we report results for both sets individually and also when the two are combined. The metric used to assess the performance is the mean average precision (MAP@3) score, which is defined as

$$\mathrm{MAP@3} = \frac{1}{N}\sum_{n=1}^{N}\sum_{i=1}^{\min\{K,3\}} P(i), \tag{4}$$

where N is the number of data samples, $K = 41$ is the number of classes, and $P(i)$ is the precision at cutoff i.

The results for the training set are shown in Table 4. The systems that are compared are the single models (log-mel configuration A only), an arithmetic-mean ensemble of the models, and the stacked ensemble described in the previous section. It can be seen that the mean ensemble performs much better than all of the single models – by almost 2 %. However, the stacked ensemble performs the best, with a MAP@3 score of 0.972. The weight-learning capability of stacking, with respect to model and class, appears to help.

In Table 5, the results for the test set are presented. We look at the 8-model stacked ensemble compared to a smaller 4-model version. In the latter, the VGG13 and CRNN architectures are omitted. The results of both systems are far superior to the competition's baseline system. Although the additional models in the 8-model version help, the difference is minor. This can be explained by the lack of diversity that the omitted models have to offer.

6. CONCLUSION

This paper described an approach to audio tagging in which the audio signals to be classified were of a diverse nature. The approach was based on our efforts in Task 2 of the DCASE 2018 challenge. The challenges of this task include audio clips of varying duration and incorrectly-labeled training examples. Our method involved preprocessing the audio, extracting log-mel feature vectors, and partitioning the feature vectors into fixed chunks to be considered as separate inputs. To train on these inputs, a number of convolutional and convolutional-recurrent neural networks were introduced. We used mixup for data augmentation. Several techniques were also used to resolve the problem of incorrect labels, including pseudo-labeling and loss function weighting. In evaluating our system on the DCASE 2018 Task 2 Kaggle private test set, we achieved a mean average precision score of 0.951, placing us in 3rd place out of 558 in the Kaggle private leaderboard.

7. ACKNOWLEDGMENT

This research was supported by the EPSRC grant EP/N014111/1, "Making Sense of Sounds", and a Research Scholarship from the China Scholarship Council (CSC), No. 201406150082. We would also like to thank Yong Xu for his contributions in the early stages of the competition and the relevant work he did in previous challenges.

8. REFERENCES

[1] J. H. L. Hansen and T. Hasan, "Speaker recognition by machines and humans: A tutorial review," *IEEE Signal Process. Mag.*, vol. 32, no. 6, pp. 74–99, 2015.

[2] M. Schedl, E. Gómez, and J. Urbano, "Music information retrieval: Recent developments and applications," *Found. Trends Inf. Retr.*, vol. 8, no. 2-3, pp. 127–261, 2014.

[3] S. Krstulović, "Audio event recognition in the smart home," in *Computational Analysis of Sound Scenes and Events*, 1st ed., T. Virtanen, M. D. Plumbley, and D. Ellis, Eds. Cham: Springer, 2018, pp. 335–371.

[4] J. P. Bello, C. Mydlarz, and J. Salamon, "Sound analysis in smart cities," in *Computational Analysis of Sound Scenes and Events*, 1st ed., T. Virtanen, M. D. Plumbley, and D. Ellis, Eds. Cham: Springer, 2018, pp. 373–397.

[5] D. Giannoulis, E. Benetos, D. Stowell, M. Rossignol, M. Lagrange, and M. D. Plumbley, "Detection and classification of acoustic scenes and events: An IEEE AASP challenge," in *IEEE Workshop Applied Signal Processing Audio and Acoustics (WASPAA)*, New Paltz, NY, 2013, pp. 1–4.

[6] E. Fonseca, M. Plakal, F. Font, D. P. W. Ellis, X. Favory, J. Pons, and X. Serra, "General-purpose tagging of Freesound audio with AudioSet labels: Task description, dataset, and baseline," *arXiv preprint arXiv:1807.09901*, 2018.

[7] E. Fonseca, J. Pons, X. Favory, F. Font, D. Bogdanov, A. Ferraro, S. Oramas, A. Porter, and X. Serra, "Freesound datasets: A platform for the creation of open audio datasets," in *Proc. 18th Int. Society Music Information Retrieval Conf. (ISMIR)*, Suzhou, China, 2017, pp. 486–493.

[8] J. F. Gemmeke, D. P. Ellis, D. Freedman, A. Jansen, W. Lawrence, R. C. Moore, M. Plakal, and M. Ritter, "Audio Set: An ontology and human-labeled dataset for audio

events," in *IEEE Int. Conf. Acoustic Speech and Signal Processing (ICASSP)*, New Orleans, LA, 2017, pp. 776–780.

[9] A. Mesaros, T. Heittola, E. Benetos, P. Foster, M. Lagrange, T. Virtanen, and M. D. Plumbley, "Detection and classification of acoustic scenes and events: Outcome of the DCASE 2016 challenge," *IEEE/ACM Trans. Audio, Speech, Language Process.*, vol. 26, no. 2, pp. 379–393, 2018.

[10] K. J. Piczak, "Environmental sound classification with convolutional neural networks," in *IEEE 25th Int. Workshop Machine Learning Signal Processing (MLSP)*, Boston, MA, 2015, pp. 1–6.

[11] H. Eghbal-zadeh, B. Lehner, M. Dorfer, and G. Widmer, "A hybrid approach with multi-channel I-vectors and convolutional neural networks for acoustic scene classification," *arXiv preprint arXiv:1706.06525*, 2017.

[12] S. Hershey, S. Chaudhuri, D. P. W. Ellis, J. F. Gemmeke, A. Jansen, C. Moore, M. Plakal, D. Platt, R. A. Saurous, B. Seybold, M. Slaney, R. Weiss, and K. Wilson, "CNN architectures for large-scale audio classification," in *IEEE Int. Conf. Acoustic Speech and Signal Processing (ICASSP)*, New Orleans, LA, 2017, pp. 131–135.

[13] E. Çakır, G. Parascandolo, T. Heittola, H. Huttunen, and T. Virtanen, "Convolutional recurrent neural networks for polyphonic sound event detection," *IEEE/ACM Trans. Audio, Speech, Language Process.*, vol. 25, no. 6, pp. 1291–1303, 2017.

[14] E. Çakır, S. Adavanne, G. Parascandolo, K. Drossos, and T. Virtanen, "Convolutional recurrent neural networks for bird audio detection," in *Proc. European Signal Processing Conf. (EUSIPCO)*, Kos, Greece, 2017, pp. 1744–1748.

[15] Y. Xu, Q. Kong, W. Wang, and M. D. Plumbley, "Large-scale weakly supervised audio classification using gated convolutional neural network," in *IEEE Int. Conf. Acoustic Speech and Signal Processing (ICASSP)*, Calgary, Canada, 2018, pp. 121–125.

[16] J. Salamon and J. P. Bello, "Deep convolutional neural networks and data augmentation for environmental sound classification," *IEEE Signal Process. Lett.*, vol. 24, no. 3, pp. 279–283, 2017.

[17] K. Xu, D. Feng, H. Mi, B. Zhu, D. Wang, L. Zhang, H. Cai, and S. Liu, "Mixup-based acoustic scene classification using multi-channel convolutional neural network," in *Advances Multimedia Information Processing (PCM)*, R. Hong, W.-H. Cheng, T. Yamasaki, M. Wang, and C.-W. Ngo, Eds., 2018, pp. 14–23.

[18] B. Frenay and M. Verleysen, "Classification in the presence of label noise: A survey," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 25, no. 5, pp. 845–869, 2014.

[19] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, "Understanding deep learning requires rethinking generalization," in *Proc. 5th Int. Conf. Learning Representations (ICLR)*, New Orleans, LA, 2017.

[20] G. Patrini, A. Rozza, A. Menon, R. Nock, and L. Qu, "Making deep neural networks robust to label noise: A loss correction approach," in *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, Honolulu, HI, 2017, pp. 2233–2241.

[21] S. Sukhbaatar, J. Bruna, M. Paluri, L. Bourdev, and R. Fergus, "Training convolutional networks with noisy labels," in *Proc. 3rd Int. Conf. Learning Representations (ICLR)*, San Diego, CA, 2015.

[22] M. Huzaifah, "Comparison of time-frequency representations for environmental sound classification using convolutional neural networks," *arXiv preprint arXiv:1706.07156*, 2017.

[23] L. Deng, J. Li, J. Huang, K. Yao, D. Yu, F. Seide, M. L. Seltzer, G. Zweig, X. He, J. D. Williams, Y. Gong, and A. Acero, "Recent advances in deep learning for speech research at microsoft," in *IEEE Int. Conf. Acoustic Speech and Signal Processing (ICASSP)*, Vancouver, Canada, 2013, pp. 8604–8608.

[24] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. 3rd Int. Conf. Learning Representations (ICLR)*, San Diego, CA, 2015.

[25] V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines," in *Proc. 27th Int. Conf. Machine Learning (ICML)*, Haifa, Israel, 2010, pp. 807–814.

[26] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. 32nd Int. Conf. Machine Learning (ICML)*, ser. Proc. Mach. Learn. Res., vol. 37, Lille, France, 2015, pp. 448–456.

[27] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Trans. Signal Process.*, vol. 45, no. 11, pp. 2673–2681, 1997.

[28] Y. N. Dauphin, A. Fan, M. Auli, and D. Grangier, "Language modeling with gated convolutional networks," in *Proc. 34th Int. Conf. Machine Learning (ICML)*, ser. Proc. Mach. Learn. Res., vol. 70, Sydney, Australia, 2017, pp. 933–941.

[29] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[30] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[31] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Advances Neural Information Processing Systems (NIPS)*, Lake Tahoe, NV, 2012, pp. 1097–1105.

[32] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," in *Proc. 6th Int. Conf. Learning Representations (ICLR)*, Vancouver, Canada, 2018.

[33] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. 3rd Int. Conf. Learning Representations (ICLR)*, San Diego, CA, 2015.

[34] D. H. Wolpert, "Stacked generalization," *Neural Netw.*, vol. 5, no. 2, pp. 241–259, 1992.

[35] K. M. Ting and I. H. Witten, "Issues in stacked generalization," *J. Artif. Int. Res.*, vol. 10, no. 1, pp. 271–289, 1999.

[36] M. J. van der Laan, E. C. Polley, and A. E. Hubbard, "Super learner," *Stat. Appl. Genet. Mol. Biol.*, vol. 6, no. 1, 2007.

DCASE 2018 CHALLENGE SURREY CROSS-TASK CONVOLUTIONAL NEURAL NETWORK BASELINE

Qiuqiang Kong[1], *Turab Iqbal*[1], *Yong Xu*[2], *Wenwu Wang*[1], *Mark D. Plumbley*[1]

Centre for Vision, Speech and Signal Processing (CVSSP), University of Surrey
[1]{q.kong, t.iqbal, w.wang, m.plumbley}@surrey.ac.uk
[2]yong.xu.ustc@gmail.com

ABSTRACT

The Detection and Classification of Acoustic Scenes and Events (DCASE) consists of five audio classification and sound event detection tasks: 1) Acoustic scene classification, 2) General-purpose audio tagging of Freesound, 3) Bird audio detection, 4) Weakly-labeled semi-supervised sound event detection and 5) Multi-channel audio classification. In this paper, we create a cross-task baseline system for all five tasks based on a convlutional neural network (CNN): a "CNN Baseline" system. We implemented CNNs with 4 layers and 8 layers originating from AlexNet and VGG from computer vision. We investigated how the performance varies from task to task with the same configuration of neural networks. Experiments show that deeper CNN with 8 layers performs better than CNN with 4 layers on all tasks except Task 1. Using CNN with 8 layers, we achieve an accuracy of 0.680 on Task 1, an accuracy of 0.895 and a mean average precision (MAP) of 0.928 on Task 2, an accuracy of 0.751 and an area under the curve (AUC) of 0.854 on Task 3, a sound event detection F1 score of 20.8% on Task 4, and an F1 score of 87.75% on Task 5. We released the Python source code of the baseline systems under the MIT license for further research.

Index Terms— DCASE 2018 challenge, convolutional neural networks, open source.

1. INTRODUCTION

Detection and classification of acoustic scenes and events (DCASE) 2018 challenge[1] is a well known IEEE challenge consists of several audio classification and sound event detection tasks. DCASE 2018 challenge consists of five tasks: In task 1, acoustic scene classification (ASC) [1], the task is to recognize the scenes where the sound is recorded, such as "street" or "park". ASC has applications in enhancing speech recognition systems and sound event detection [2]. Task 1 includes a matching device ASC subtask and a mismatching device ASC subtask. In task 2, general-purpose audio tagging of Freesound, [3] the task is to classify an audio clip to a pre-defined class, such as "flute" or "applause". Task 2 has applications in recognizing a wide range of sound events in real world and is useful for information retrieval. In task 3, bird audio detection, [4], the task is to detect the presence or the absence of birds in an audio clip. This could be used for automatic wildlife monitoring and audio library management. An important goal of Task 3 is to design a classification system which can generalize to new conditions. In task 4, weakly labeled semi-supervised sound event detection (SED) [5], the task is to detect the onset the offset times of sound events where only weak labeled audio and unlabeled audio is available for training. Task 4 can be used for monitoring public security and used for abnormal sound detection. In task 5, the multi-channel audio classification [6], the task is to use multi-channel recordings to identify the human activities at home.

The first DCASE challenge was the DCASE 2013 challenge [7], with only an audio classification and a sound event detection tasks. The DCASE 2016 challenge [8] consisted of four tasks including: 1) ASC, 2) SED in synthetic audio, 3) SED in real audio and 4) domestic audio tagging. The DCASE 2017 challenge [9] updated the domestic audio tagging task to a large-scale weakly labeled audio tagging task. The DCASE challenge series provide public datasets for investigating audio related tasks. One recent dataset for DCASE challenges is the AudioSet dataset [10]. Task 4 of both DCASE 2017 and 2018 challenge were subsets of AudioSet.

Convolutional neural networks (CNNs) have achieved state-of-the-art performance in image classification [11, 12]. In this paper, we investigate how different CNNs including CNN with 4 layers originated from AlexNet [11] and CNN with 8 layers originated from VGG [12] perform on Task 1 to 5 of DCASE 2018. We apply the same configurations of CNNs across all task 1 to 5 to fairly compare the relative performance across different tasks. Using the same CNN model, the performance on Task 1 to 5 varies, which indicates the difficulty of the tasks varies. The experiments show that Task 4 sound event detection is more difficult than Task 1 acoustic scene classification than Task 3 bird audio detection than Task 2 general-purpose audio tagging of Freesound and Task 5 domestic multi-channel audio tagging.

We open source the Python code for all of Task 1 - 5 of DCASE 2018 challenge under MIT license. The source code contains the implementation of CNNs with 4 layers and 8 layers. In complementary to the source code published by the organizer [1], we investigated that CNNs with more layers perform better in all of Task 2 - 5 in DCASE 2018 challenge except Task 1.

This paper is organized as follows, Section 2 introduces related works. Section 3 introduces CNNs. Section 4 shows experimental results. Section 5 concludes and forecasts our work.

2. RELATED WORKS

Manually-selected features such as mel frequency cepstrum coefficients (MFCC) [13], the constant Q transform (CQT) [14], and I-vectors [15] have been used as audio features. Recently, mel spectrograms [16] have been widely used as features when using neural networks as classifiers. Mixture Gaussian models (GMMs) [17] and hidden Markov models (HMMs) [18] have been used to model audio scenes and sound events. Non-negative matrix factor-

[1]http://dcase.community/

Table 1: Configurations of CNN4 and CNN8

feature map size	CNN4	CNN8
$T \times 64$	log mel spectrogram	
$T/2 \times 32$	$5 \times 5, 64$	$\begin{bmatrix} 3 \times 3, \text{BN} \\ 3 \times 3, \text{BN} \end{bmatrix}, 64$
	2×2, max pooling	
$T/4 \times 16$	$5 \times 5, 128$	$\begin{bmatrix} 3 \times 3, \text{BN} \\ 3 \times 3, \text{BN} \end{bmatrix}, 128$
	2×2, max pooling	
$T/8 \times 8$	$5 \times 5, 256$	$\begin{bmatrix} 3 \times 3, \text{BN} \\ 3 \times 3, \text{BN} \end{bmatrix}, 256$
	2×2, max pooling	
$T/16 \times 4$	$5 \times 5, 512$	$\begin{bmatrix} 3 \times 3, \text{BN} \\ 3 \times 3, \text{BN} \end{bmatrix}, 512$
	2×2, max pooling	
1×1	Global max pooling	
	Classes num. fc, sigmoid or softmax	
Parameters	4,309,450	4,691,274

ization (NMFs) [19] are methods to learn a set of bases to represent the audio. Recently, deep neural networks have been introduced to audio classification and sound event detection. For example, fully-connected neural networks have been applied to DCASE 2016 challenges [20] and DCASE 2017 challenges [21]. CNNs have achieved the state-of-the-art performance in audio classification and sound event detection [22, 16, 23]. Convolutional recurrent neural networks (RNNs) [24, 25] have been used to model the temporal information of sound events. Attention neural networks have been proposed to focus on sound events [26] from weakly-labelled data [27]. Generative adversarial networks (GANs) have been applied to improve the robustness of audio classification classifiers [28].

3. CONVOLUTIONAL NEURAL NETWORKS

CNNs, such as AlexNet [11] and VGG [12], have achieved state-of-the-art performance in image classification [11, 12]. A CNN consists of several convolutional layers followed by fully-connected layers. Each convolutional layer consists of filters to convolve with the output from the previous convolutional layer. The filters can capture local patterns in feature maps, such as edges in lower layers and complex profiles in higher layers [12]. In this work, we adopt AlexNet with 4 layers and VGG with 8 layers as models, which we call CNN4 and CNN8. CNN4 consists of 4 convolutional layers and the filter size of each convolutional layer is 5×5 [11]. CNN8 consists of 8 layers and the filter size of each convolutional layer is 3×3 [12]. We apply batch normalization (BN) after each convolutional layer to stabilize training [29] followed by a rectifier (ReLU) nonlinearity. We then apply a global max pooling (GMP) operation on the feature maps of the last convolutional layer [16] to summarize the feature maps to a vector. GMP can max out the time and frequency information of sound events in a spectrogram, so it is invariant to time or frequency shift. Finally, a fully-connected layer is applied on the summarized vector followed by a sigmoid or softmax nonlinearity to output the probabilities of the audio classes. The configurations of CNN4 and CNN8 are summarized in Table 1.

Table 2: Task 1 acoustic scene classification class-wise accuracy of subtask A and B of development dataset.

Scene label	SUBTASK A			SUBTASK B		
	CNN [1]	CNN4	CNN8	CNN [1]	CNN4	CNN8
Airport	0.729	0.743	0.709	0.725	0.612	0.667
Bus	0.629	0.607	0.649	0.783	0.695	0.723
Metro	0.512	0.690	0.686	0.206	0.500	0.417
Metro station	0.554	0.687	0.741	0.328	0.472	0.584
Park	0.791	0.855	0.839	0.592	0.834	0.861
public square	0.404	0.486	0.472	0.247	0.361	0.389
Shopping mall	0.496	0.642	0.631	0.611	0.778	0.778
Street, pedestrian	0.500	0.583	0.567	0.208	0.333	0.361
Street, traffic	0.805	0.874	0.886	0.664	0.750	0.778
Tram	0.551	0.590	0.621	0.197	0.417	0.389
Average	0.597	0.676	**0.680**	0.456	**0.575**	0.572
Public LB	-	0.693	**0.707**	-	**0.578**	0.568
Private LB	-	0.628	**0.630**	-	0.615	**0.672**
Evaluation	-	0.697	**0.704**	-	0.588	**0.596**

4. EXPERIMENTS

We open source the Python code of the CNN baseline systems of DCASE 2018 challenge Task 1 - 5 source here[2][3][4][5][6]. We convert all stereo audio to mono for Task 1 - 5 for building the baseline system. We extract the spectrograms and apply log mel filter banks on the spectrograms followed by logarithm operation. We choose the number of the mel filter banks as 64 because it is a power of two which can be divided by two in max pooling layers. The mel filter bank has a cut off frequency of 50 Hz. The log mel spectrograms are standarized by subtracting the mean and dividing the standard deviation along mel frequency bins. The same configuration of CNN4 and CNN8 are applied on Task 1 - 5. We use Adam optimizer [30] with a learning rate of 0.001 and the learning rate is reduced by multiplying 0.9 after every 200 iterations training. A batch size of 128 is used for Task 1, 2, 3 and 5 and a batch size of 32 is used for Task 4 to sufficiently use the GPU with 12 GB memory in training. We trained the model for 5000 iterations for all of the five tasks. The training takes 60 ms and 200 ms per iteration on a Titan X GPU for CNN4 and CNN8, respectively. The results of Task 1 - 5 are shown in the following subsections.

4.1. Task 1: Acoustic scene classification

Task 1 acoustic scene classification [1] is a task to classify an audio recording to a predefined class that characterize the environment in which it was recorded. The 10 predefined classes are listed in Table 2. There are 10080 10-second audio clips in the development dataset, including 8640, 720 and 720 audio clips recorded with device A, B and C. Task 1 has three subtasks. Subtask A is matching

[2]https://github.com/qiuqiangkong/dcase2018_task1
[3]https://github.com/qiuqiangkong/dcase2018_task2
[4]https://github.com/qiuqiangkong/dcase2018_task3
[5]https://github.com/qiuqiangkong/dcase2018_task4
[6]https://github.com/qiuqiangkong/dcase2018_task5

Table 3: Task 2 audio tagging accuracy and MAP@3.

	Accuracy		MAP@3	
	CNN4	CNN8	CNN4	CNN8
Fold 1	0.858	0.897	0.900	0.930
Fold 2	0.824	0.875	0.870	0.912
Fold 3	0.862	0.903	0.901	0.934
Fold 4	0.861	0.904	0.904	0.935
Average	0.851	**0.895**	0.894	**0.928**
Public LB	-	-	0.885	**0.920**
Private LB	-	-	0.862	**0.903**

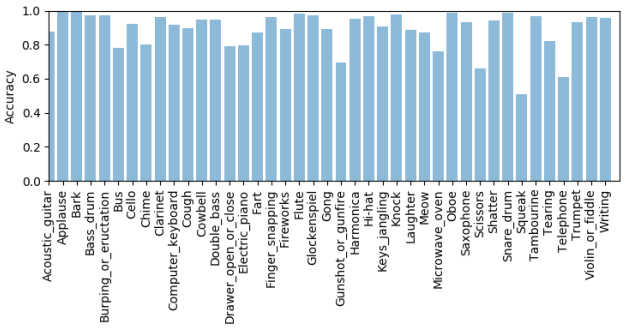


Figure 1: Task 2 audio tagging class-wise accuracy.

device classification. Subtask B is mismatching device classification. Subtask C is matching device classification with external data and has the same evaluation data as subtask A.

Table 2 shows the accuracy of subtask A and subtask B. In [1] a two layer CNN with a dense connected layer is used as a baseline model. In development dataset of subtask A, CNN4 and CNN8 achieve similar accuracy of 0.676 and 0.680 respectively, outperforming the two layers CNN of 0.597 [1]. In subtask B, CNN4 and CNN8 achieve similar accuracy of 0.575 and 0.572, respectively, outperforming the two layers CNN of 0.456 [1]. The subtask B mismatching device classification is around 10% which is worse than the subtask A matching device classification in absolute value. Table 2 also shows the public leaderboard (LB), private LB and final evaluation result. We did not explore the subtask C with external data.

4.2. Task 2: General-purpose audio tagging of Freesound content with AudioSet labels

Task 2 audio tagging [3] is a task to classify an audio clip to one of 41 predefined classes such as "oboe" and "applause". The duration of the audio samples range from 300 ms to 30 s due to the diversity of the sound categories. The development dataset contains 9473 audio clips. We pad or split the log mel spectrograms of audio clips to 2 s log mel spectrograms as the input to a CNN. We split the development dataset to four validation folds and only use 3710 manually verified audio clips for validation. Table 3 shows the accuracy and the mean average precision (MAP) [3] on the four folds and their average statistics. In development dataset, CNN8 achieves an average accuracy of 0.895 and a MAP@3 of 0.928, outperforming CNN4 network of 0.851 and 0.894, respectively. Figure 1 shows the averaged 4 folds class-wise accuracy of Task 2. Sound classes such as "applause" and "bark" have 100% classification accuracy but some classes such as "squeak" and "telephone" have accuracy of only 50% - 60%. Table 3 shows the MAP@3 of the private leaderboard is approximately 2% worse than the development and the public leaderboard.

Table 4: Task 3 bird audio detection accuracy and AUC.

	Accuracy		AUC	
validation dataset	CNN4	CNN8	CNN4	CNN8
freefield1010	0.551	0.630	0.645	0.799
warblrb10k	0.692	0.867	0.799	0.882
BirdVox-DCASE-20K	0.678	0.801	0.808	0.882
Average	0.640	**0.766**	0.751	**0.854**
Leaderboard	-	-	**0.850**	0.847
Evaluation	-	-	0.748	**0.809**

4.3. Task 3: Bird audio detection

Task 3 bird audio detection [4] is a task to predict the presence or the absence of birds in a 10-second audio clip. One challenge of this task is to design a system that is able to generalize to new conditions. That is, a system trained on one dataset should generalize well to another dataset. The development dataset consists of freefield1010 with 7690 audio clips, warblrb10k with 8000 audio clips and BirdVox-DCASE-20K with 20000 audio clips. We train on two datasets and evaluate on the other hold out dataset. Table 4 shows the accuracy and the area under the curve (AUC) [4] of CNN4 and CNN8. In development dataset, CNN8 achieves an accuracy and an AUC of 0.766 and 0.854, outperforming CNN4 of 0.640 and 0.751, respectively. The result in Table 4 shows the classification of freefield1010 dataset is more difficult than warblrb10k and BirdVox-DCASE-20K dataset.

Table 5: Task 4 audio tagging AUC and sound event detection F1 score.

	AT (AUC)		SED1 (F1)		SED2 (F1)	
Class	CNN4	CNN8	CNN4	CNN8	CNN4	CNN8
Speech	0.889	0.936	0.0%	0.0%	16.9%	**22.5%**
Dog	1.000	1.000	2.6%	2.5%	8.3%	**14.3%**
Cat	0.980	0.991	3.4%	3.5%	**10.3%**	7.2%
Alarm/bell	0.964	0.975	4.2%	4.0%	12.5%	**20.7%**
Dishes	0.835	0.898	0.0%	0.0%	0.0%	**3.6%**
Frying	0.945	0.939	45.5%	**54.5%**	2.1%	0.0%
Blender	0.839	0.883	18.9%	**27.1%**	8.3%	7.3%
Running water	0.930	0.943	11.8%	**11.9%**	7.9%	3.1%
Vacuum cleaner	0.972	0.956	57.6%	**61.3%**	9.4%	2.6%
Electronic shaver	0.944	0.957	**45.0%**	43.5%	18.9%	16.3%
Average	0.930	0.948	18.9%	**20.8%**	9.5%	9.8%
Evaluation	-	-	16.7%	**18.6%**	-	-

Table 6: Task 5 multi-channel audio tagging F1 score.

		CNN4 (F1 score)					CNN8 (F1 score)				
Scene label	Baseline	Fold 1	Fold 2	Fold 3	Fold 4	Average	Fold 1	Fold 2	Fold 3	Fold 4	Average
Absence	85.4%	86.4%	90.5%	78.5%	89.9%	86.3%	90.5%	92.2%	80.5%	89.9%	88.3%
Cooking	95.1%	96.2%	94.7%	93.0%	96.6%	95.1%	98.0%	96.3%	93.8%	96.3%	96.1%
Dishwashing	76.7%	77.8%	68.6%	75.8%	80.2%	75.6%	83.3%	71.2%	76.0%	85.8%	79.1%
Eating	83.6%	79.7%	75.7%	85.4%	91.2%	82.3%	85.2%	85.1%	88.5%	94.5%	88.3%
Other	44.8%	43.3%	55.2%	56.9%	60.2%	53.9%	54.3%	54.5%	51.4%	62.2%	55.6%
Social activity	93.9%	95.5%	88.1%	90.2%	98.5%	93.1%	98.4%	90.1%	93.7%	99.3%	95.4%
Vacuum cleaner	99.3%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	99.4%	100.0%	100.0%	99.9%
Watching TV	99.6%	99.6%	99.7%	97.5%	100.0%	99.2%	99.8%	99.9%	99.0%	99.9%	99.7%
Working	82.0%	85.3%	86.3%	79.4%	90.5%	85.4%	88.7%	89.3%	81.4%	90.2%	87.4%
Average	84.5%	84.9%	84.3%	84.1%	89.7%	**85.7%**	88.7%	86.5%	84.9%	90.9%	**87.8%**
Eval. Unknown mic.	83.1%	-	-	-	-	82.4%	-	-	-	-	**83.2%**
Eval. dev. mic.	85.0%	-	-	-	-	86.2%	-	-	-	-	**87.6%**

Furthermore, an AUC of 0.809 is achieved in evaluation dataset using CNN8.

4.4. Task 4: Large-scale weakly labeled semi-supervised sound event detection in domestic environments

Task 4 is a weakly labeled semi-supervised sound event detection task [5] to predict both the onset and offset of sound events. There are 10 audio classes in Task 4, for example "speech" and "dog". An audio clip can be assigned to one or more labels. The development dataset consists of 1578 weakly labeled audio clips, 14412 unlabeled in domain audio clips and 39999 unlabeled out domain audio clips. Each audio clip has a duration of 10 seconds. We only use the 1578 weakly labeled audio clips for training our systems. Different from Task 1, 2, 3 and 5, to remain the time resolution of feature maps in time axis, max pooling operations are only applied along the frequency axis but not the time axis. In training, we average out the time axis and apply a fully connected layer to predict the clip-wise labels. In inference, we do not apply the average of time axis to remain frame-wise labels. Table 5 shows CNN8 achieves an AUC of 0.948 in audio tagging, outperforming CNN4 of 0.930. In sound event detection, system SED1 uses the audio tagging result as the sound event detection result. The onset and offset times are filled with 0 s and 10 s. System SED2 applies thresholds to the frame-wise predictions to detect sound events. The high threshold and the low threshold are set as 0.8 and 0.2, respectively. Sound events such as "Frying", "Blender" have higher F1 score with SED1. Sound event such as "Speech", "Dog", "Cat" have higher F1 score with SED2. In development dataset, SED1 and SED2 achieve average F1 scores of 20.8% and 9.8%, respectively. In evaluation, a F1 score of 18.6% is achieved using CNN8 and system SED1.

4.5. Task 5: Monitoring of domestic activities based on multi-channel acoustics

Task 5 multi-channel audio tagging [6] is a task to classify the domestic activities with multi-channel acoustic recordings. The target of Task 5 is to research how the multi-channel information will help the audio tagging task. The development dataset of Task 5 consists of 72984 10-second audio clips. The audio classes including "Cooking" and "Eating", for example. The multi-channel audio clips are converted to single channel audio clips to build the baseline system. Table 6 shows in development dataset the CNN8 achieves a F1 score of 87.75%, outperforming CNN4 network of 85.73%. In Evaluation data with unknown microphone a F1 score of 83.2% is achieved using CNN8 model. With unkown development microphone, a F1 score of 87.6% is achieved.

5. CONCLUSION

In this paper, we investigated the performance of convolutional neural networks (CNNs) with 4 layers and 8 layers on Task 1 to 5 of DCASE 2018. We show the difficulties of the tasks varies. Task 4 sound event detection is more difficult than Task 1 acoustic scene classification than Task 3 bird audio detection than Task 2 general-purpose audio tagging of Freesound and Task 5 domestic multi-channel audio tagging. We show CNN with 8 layers performs better than CNN with 4 layers in Task 2 to 5. In Task 1, CNN with 8 layers and 4 layers perform similar. With CNN8, we achieves an accuracy of 0.680 on Task 1, a mean average precision (MAP) of 0.928 on Task 2, an area under the curve (AUC) of 0.854 on Task 3, a sound event detection F1 score of 20.8% on Task 4 and a F1 score of 87.75% on Task 5. In future, we will explore more CNN structures on Task 1 to 5 of DCASE 2018 challenge. We released the Python source code of the baseline systems under the MIT license for further research.

6. ACKNOWLEDGEMENT

This research was supported by EPSRC grant EP/N014111/1 "Making Sense of Sounds" and a Research Scholarship from the China Scholarship Council (CSC) No. 201406150082.

7. REFERENCES

[1] A. Mesaros, T. Heittola, and T. Virtanen, "A multi-device dataset for urban acoustic scene classification," *arXiv preprint arXiv:1807.09840*, 2018.

[2] D. Barchiesi, D. Giannoulis, D. Stowell, and M. D. Plumbley, "Acoustic scene classification," *arXiv preprint arXiv:1411.3715*, 2014.

[3] E. Fonseca, M. Plakal, F. Font, D. P. W. Ellis, X. Favory, J. Pons, and X. Serra, "General-purpose tagging of freesound audio with audioset labels: Task description, dataset, and baseline," *arXiv preprint arXiv:1807.09902*, 2018.

[4] D. Stowell, Y. Stylianou, M. Wood, H. Pamuła, and H. Glotin, "Automatic acoustic detection of birds through deep learning: the first bird audio detection challenge," *arXiv preprint arXiv:1807.05812*, 2018.

[5] R. Serizel, T. Nicolas, H. Eghbal-Zadeh, and A. P. Shah, "Large-scale weakly labeled semi-supervised sound event detection in domestic environments," *https://hal.inria.fr/hal-01850270*, 2018.

[6] G. Dekkers, S. Lauwereins, B. Thoen, M. W. Adhana, H. Brouckxon, T. van Waterschoot, B. Vanrumste, M. Verhelst, and P. Karsmakers, "The SINS database for detection of daily activities in a home environment using an acoustic sensor network," in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2017 Workshop (DCASE2017)*, Munich, Germany, November 2017, pp. 32–36.

[7] D. Giannoulis, E. Benetos, D. Stowell, M. Rossignol, M. Lagrange, and M. D. Plumbley, "Detection and classification of acoustic scenes and events: An IEEE AASP challenge," in *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*. IEEE, 2013, pp. 1–4.

[8] A. Mesaros, T. Heittola, and T. Virtanen, "TUT database for acoustic scene classification and sound event detection," in *Signal Processing Conference (EUSIPCO)*. IEEE, 2016, pp. 1128–1132.

[9] A. Mesaros, T. Heittola, A. Diment, B. Elizalde, A. Shah, E. Vincent, B. Raj, and T. Virtanen, "DCASE 2017 challenge setup: Tasks, datasets and baseline system," in *DCASE 2017-Workshop on Detection and Classification of Acoustic Scenes and Events*, 2017.

[10] J. F. Gemmeke, D. P. W. Ellis, D. Freedman, A. Jansen, W. Lawrence, R. C. Moore, M. Plakal, and M. Ritter, "Audio set: An ontology and human-labeled dataset for audio events," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017, pp. 776–780.

[11] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems (NIPS)*, 2012, pp. 1097–1105.

[12] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[13] D. Li, I. K. Sethi, N. Dimitrova, and T. McGee, "Classification of general audio data for content-based retrieval," *Pattern Recognition Letters*, vol. 22, no. 5, pp. 533–544, 2001.

[14] V. Bisot, R. Serizel, S. Essid, and G. Richard, "Supervised nonnegative matrix factorization for acoustic scene classification," *IEEE AASP Challenge on Detection and Classification of Acoustic Scenes and Events (DCASE)*, 2016.

[15] H. Eghbal-Zadeh, B. Lehner, M. Dorfer, and G. Widmer, "CP-JKU submissions for DCASE-2016: A hybrid approach using binaural i-vectors and deep convolutional neural networks," *IEEE AASP Challenge on Detection and Classification of Acoustic Scenes and Events (DCASE)*, 2016.

[16] K. Choi, G. Fazekas, and M. Sandler, "Automatic tagging using deep convolutional neural networks," *arXiv preprint arXiv:1606.00298*, 2016.

[17] D. Stowell, D. Giannoulis, E. Benetos, M. Lagrange, and M. D. Plumbley, "Detection and classification of acoustic scenes and events," *IEEE Transactions on Multimedia*, vol. 17, no. 10, pp. 1733–1746, 2015.

[18] A. Mesaros, T. Heittola, A. Eronen, and T. Virtanen, "Acoustic event detection in real life recordings," in *Signal Processing Conference*. IEEE, 2010, pp. 1267–1271.

[19] A. Mesaros, T. Heittola, O. Dikmen, and T. Virtanen, "Sound event detection in real life recordings using coupled matrix factorization of spectral representations and class activity annotations," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2015, pp. 151–155.

[20] Q. Kong, I. Sobieraj, W. Wang, and M. D. Plumbley, "Deep neural network baseline for dcase challenge 2016," *Proceedings of DCASE 2016*, 2016.

[21] J. Li, W. Dai, F. Metze, S. Qu, and S. Das, "A comparison of deep learning methods for environmental sound," *arXiv preprint arXiv:1703.06902*, 2017.

[22] S. Hershey, S. Chaudhuri, D. P. Ellis, J. F. Gemmeke, A. Jansen, R. C. Moore, M. Plakal, D. Platt, R. A. Saurous, B. Seybold *et al.*, "CNN architectures for large-scale audio classification," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017, pp. 131–135.

[23] Y. Aytar, C. Vondrick, and A. Torralba, "Soundnet: Learning sound representations from unlabeled video," in *Advances in Neural Information Processing Systems (NIPS)*, 2016, pp. 892–900.

[24] E. Cakir, G. Parascandolo, T. Heittola, H. Huttunen, T. Virtanen, E. Cakir, G. Parascandolo, T. Heittola, H. Huttunen, and T. Virtanen, "Convolutional recurrent neural networks for polyphonic sound event detection," *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, vol. 25, no. 6, pp. 1291–1303, 2017.

[25] H. Lim, J. Park, K. Lee, and Y. Han, "Rare sound event detection using 1D convolutional recurrent neural networks," DCASE2017 Challenge, Tech. Rep., 2017.

[26] Y. Xu, Q. Kong, W. Wang, and M. D. Plumbley, "Large-scale weakly supervised audio classification using gated convolutional neural network," *arXiv preprint arXiv:1710.00343*, 2017.

[27] A. Kumar and B. Raj, "Audio event detection using weakly labeled data," in *Proceedings of the 2016 ACM on Multimedia Conference*. ACM, 2016, pp. 1038–1047.

[28] S. Mun, S. Park, D. K. Han, and H. Ko, "Generative adversarial network based acoustic scene training set augmentation and selection using svm hyper-plane," *Proc. DCASE*, pp. 93–97, 2017.

[29] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[30] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

Tampereen teknillinen yliopisto
PL 527
33101 Tampere

Tampere University of Technology
P.O.B. 527
FI-33101 Tampere, Finland

ISBN 978-952-15-4262-6